MPC5125 Microcontroller Reference Manual

Devices Supported:
MPC5125

Document Number: MPC5125RM
Rev 3
10/2011

This page is intentionally left blank

Chapter 1
Overview

1.1 Introduction . . . 1
1.2 Chip-Level Features . . . 2
1.3 Module Features . . . 3
1.3.1 e300 Processor Core . . . 3
1.3.2 Display Interface Unit (DIU) . . . 3
1.3.3 USB Controller . . . 3
1.3.4 Direct Memory Access (DMA) Controller . . . 3
1.3.5 DDR SDRAM Memory Controller . . . 4
1.3.6 32 KB On-chip SRAM . . . 4
1.3.7 Fast Ethernet Controller (FEC) . . . 4
1.3.8 NAND Flash Interface . . . 4
1.3.9 Local Plus Bus (LPC) Interface . . . 4
1.3.10 Secure Digital Host Controller (SDHC) . . . 4
1.3.11 Controller Area Network (CAN) . . . 5
1.3.12 Integrated Circuit Communication (I^2C) . . . 5
1.3.13 Programmable Serial Controller (PSC) . . . 5
1.3.14 J1850 Byte Data Link Controller (BDLC) Interface . . . 5
1.3.15 General Purpose I/Os (GPIO) . . . 5
1.3.16 On-chip Real-time Clock (RTC) . . . 5
1.3.17 IC Identification Module (IIM) . . . 6
1.3.18 On-chip Temperature Sensor . . . 6
1.3.19 Power Modes . . . 6
1.3.20 System Timers . . . 6
1.3.21 IEEE 1149.1 Compliant JTAG Boundary Scan . . . 6
1.4 Developer Environment . . . 6

Chapter 2
System Configuration and Memory Map (XLBMEN + Mem Map)

2.1 Introduction . . . 7
2.2 Memory Map and Register Definition . . . 7
2.2.1 Local Memory Map Overview and Example . . . 7
2.2.2 Address Translation and Mapping . . . 8
2.2.3 Window into Configuration Space . . . 8
2.2.4 Local Access Windows . . . 9
2.2.5 Local Access Register Memory Map . . . 9
2.2.6 Overlap of Local Access Windows . . . 20
2.2.7 Configuring Local Access Windows . . . 20
2.3 System Configuration . . . 21
2.3.1 System Configuration Register Memory Map . . . 21

Chapter 3
Signal Descriptions

3.1 Introduction . . . 25
3.1.1 Signals Overview . . . 25
3.2 Signal States During Reset . . . 57

Chapter 4
Reset

4.1 Introduction . . . 61
4.2 Power-On Initialization ($\overline{\text{PORESET}}$) . . . 63
4.3 $\overline{\text{HRESET}}$ Flow . . . 63
4.3.1 Sources . . . 63
4.3.2 Impacts . . . 63
4.4 $\overline{\text{SRESET}}$ Flow . . . 64
4.4.1 Sources . . . 64
4.4.2 Impacts . . . 64
4.5 Reset Configuration Word (RST_CONF) . . . 64
4.5.1 BMS Operation . . . 66
4.5.2 RTC at Reset . . . 66
4.5.3 JTAG Reset . . . 66
4.5.4 Boot Vector Selection . . . 66
4.5.5 Boot Memory Interface Selection . . . 66
4.5.6 LPC Initialization Sequence . . . 67
4.5.7 NFC Initialization Sequence . . . 67
4.6 Memory Map . . . 68
4.6.1 Reset Configuration Word Low (RCWL) Register . . . 69
4.6.2 Reset Configuration Word High (RCWH) Register . . . 69
4.6.3 Reset Status Register (RSR) . . . 70
4.6.4 Reset Mode Register (RMR) . . . 72
4.6.5 Reset Protection Register (RPR) . . . 73
4.6.6 Reset Control Register (RCR) . . . 73
4.6.7 Reset Control Enable Register (RCER) . . . 74

Chapter 5
Clocks and Low-Power Modes

5.1 Introduction . . . 75
5.2 System Clock Generation . . . 75
5.2.1 Peripheral Clock Domains . . . 76
5.2.2 System Oscillator Disable . . . 77
5.2.3 PSC Clock Generation . . . 77
5.2.4 MSCAN Clock Generation . . . 78
5.2.5 OUT Clock Generation . . . 78
5.2.6 RTC Clock Generation . . . 78
5.2.7 System PLL lock detection . . . 79

5.2.8 System PLL and e300 PLL . . . 79
5.3 Clock Control Module . . . 80
5.3.1 Memory Map/Register Definition . . . 80

Chapter 6
Byte Data Link Controller (BDLC)

6.1 Introduction . . . 111
6.1.1 Features . . . 113
6.2 External Signal Description . . . 113
6.3 Memory Map and Register Definition . . . 113
6.3.1 Memory Map . . . 113
6.3.2 Register Summary . . . 114
6.4 Functional Description . . . 129
6.4.1 J1850 Frame Format . . . 129
6.4.2 J1850 VPW Symbols . . . 131
6.4.3 MUX Interface . . . 143
6.4.4 Protocol Handler . . . 145
6.4.5 Transmitting a Message . . . 147
6.4.6 Receiving A Message . . . 152
6.4.7 Transmitting an In-Frame Response (IFR) . . . 156
6.4.8 Receiving An In-Frame Response (IFR) . . . 165
6.4.9 Special BDLC Module Operations . . . 167
6.5 Initialization Information . . . 169
6.5.1 Initializing the Configuration Bits . . . 169
6.5.2 Exiting Loopback Mode and Enabling the BDLC Module . . . 170
6.5.3 Enabling BDLC Interrupts . . . 170

Chapter 7
CPU e300 Core Power Architecture

7.1 Introduction . . . 173
7.2 e300c4 Processor Core Functional Overview . . . 173
7.3 e300c4 Core Reference Manual . . . 174
7.4 Unsupported e300c4 Core Features . . . 174
7.4.1 Instructions . . . 174
7.4.2 CSB Parity . . . 174

Chapter 8
CSB Arbiter and Bus Monitor

8.1 Introduction . . . 175
8.1.1 Features . . . 175
8.2 Memory Map/Register Definition . . . 176
8.2.1 Register Descriptions . . . 176
8.3 Functional Description . . . 186
8.3.1 Arbitration Policy . . . 186

8.3.2 Bus Error Detection . . . 189
8.4 Initialization/Applications Information . . . 192
8.4.1 Initialization Sequence . . . 192
8.4.2 Error Handling Sequence . . . 193

Chapter 9
Direct Memory Access (DMA)

9.1 Introduction . . . 195
9.1.1 Features . . . 195
9.2 Memory Map and Register Definition . . . 195
9.2.1 Register Descriptions . . . 207
9.3 Initialization/Application Information . . . 230
9.3.1 DMA Initialization . . . 230
9.3.2 DMA Programming Errors . . . 230
9.3.3 DMA Arbitration Mode Considerations . . . 231
9.3.4 DMA Transfer . . . 232
9.3.5 TCD Status . . . 235
9.3.6 Channel Linking . . . 236
9.3.7 Dynamic Programming . . . 237

Chapter 10
Display Interface Unit (DIU)

10.1 Introduction . . . 239
10.1.1 Features . . . 239
10.1.2 Modes of Operation . . . 239
10.2 External Signal Description . . . 240
10.3 Memory Map and Register Definition . . . 240
10.3.1 Memory Map . . . 240
10.3.2 Register Summary . . . 242
10.3.3 Register Descriptions . . . 246
10.4 Functional Description . . . 263
10.4.1 Area Descriptor . . . 263
10.4.2 Area Descriptor Format . . . 265
10.4.3 Pixel Structure . . . 272
10.4.4 Pixel Format Conversion . . . 273
10.4.5 Alpha Blending . . . 274
10.4.6 Chroma Keying . . . 275
10.4.7 Gamma Correction . . . 275
10.4.8 Cursor . . . 276
10.4.9 Writeback Operation . . . 278
10.4.10 Color Bar Generation . . . 279
10.4.11 Interrupt Generation . . . 279
10.4.12 Dynamic Priority Generation . . . 280
10.4.13 Display Signal Timing . . . 281

10.5 Initialization/Application Information 282
10.5.1 DIU Initialization 282
10.5.2 Controlling DIU Planes after the DIU is Enabled 283
10.5.3 Synchronizing with the Host (CPU) 283
10.5.4 Recovering from Parameter Error 284
10.5.5 Recovering from Underrun Error 284

Chapter 11
DRAM Controller

11.1 Introduction 285
11.1.1 Overview 285
11.2 Features 286
11.3 Memory Map and Register Definition 288
11.3.1 Memory Map 288
11.3.2 Register Descriptions 289
11.4 Functional Description 304
11.4.1 Interfacing with the DRAM 304
11.4.2 Programming DRAM Device Internal Configuration Register 305
11.4.3 DRAM Command Engine 305
11.4.4 Write Buffer 306
11.4.5 Timing Manager 306
11.4.6 DRAM Read Block and DRAM Write Block 306
11.4.7 Bus Interface 306

Chapter 12
Multi-port DRAM Controller Priority Manager

12.1 Introduction 309
12.1.1 Features 310
12.2 Bus Connections 310
12.3 Memory Map and Register Definition 310
12.3.1 Memory Map 310
12.3.2 Register Descriptions 313
12.4 Functional Description 327
12.4.1 Description of Operation — Overview 327
12.4.2 Block Diagram 327
12.4.3 Congestion Detector 329

Chapter 13
External Memory Bus (EMB)

13.1 Introduction 331
13.1.1 Overview 331
13.1.2 Features 331
13.2 Functional Description 331
13.2.1 EMB Mux 331

Chapter 14
Fast Ethernet Controller (FEC)

14.1 Introduction . . . 333
14.1.1 FEC Top Level . . . 333
14.1.2 Features . . . 335
14.1.3 Modes of Operation . . . 336
14.2 External Signal Description (Off Chip) . . . 337
14.3 Memory Map and Register Definition . . . 338
14.3.1 Overview . . . 338
14.3.2 Top-Level Module Memory Map . . . 338
14.3.3 Detailed Memory Map—Control/Status Registers . . . 339
14.3.4 MIB Block Counters Memory Map . . . 341
14.3.5 Register Descriptions . . . 343
14.4 Initialization Information . . . 365
14.4.1 Initialization Prior to Asserting ETHER_EN . . . 365
14.5 Buffer Descriptors . . . 367
14.5.1 Driver/DMA Operation with Buffer Descriptors . . . 367
14.5.2 Ethernet Receive Buffer Descriptor (RxBD) . . . 369
14.5.3 Ethernet Transmit Buffer Descriptor . . . 371
14.6 Network Interface Options . . . 372
14.6.1 FEC Frame Transmission . . . 374
14.6.2 FEC Frame Reception . . . 375
14.6.3 Ethernet Address Recognition . . . 376
14.6.4 Full-Duplex Flow Control . . . 381
14.6.5 Inter-Packet Gap Time . . . 382
14.6.6 Collision Handling . . . 382
14.6.7 MII Internal and External Loopback . . . 382
14.6.8 RMII Loopback . . . 383
14.6.9 RMII Echo . . . 383
14.6.10 Ethernet Error-Handling Procedure . . . 383
14.6.11 Transmission Errors . . . 383
14.6.12 Reception Errors . . . 384

Chapter 15
General Purpose Timers (GPT)

15.1 Introduction . . . 387
15.1.1 Modes of Operation . . . 387
15.1.2 Detailed Signal Descriptions . . . 388
15.2 Memory Map and Register Definition . . . 388
15.2.1 Register Descriptions . . . 391
15.3 Functional Description . . . 397
15.3.1 Input Capture Mode . . . 397
15.3.2 Changing Sub-Modes . . . 398
15.3.3 Output Compare . . . 399

15.3.4 Force Output Low Immediately . . . 399
15.3.5 Output Pulse High . . . 399
15.3.6 Output Pulse Low . . . 400
15.3.7 Output Toggle . . . 400
15.3.8 Pulse Width Modulation . . . 401
15.3.9 Simple GPIO . . . 404

Chapter 16
General Purpose I/O (GPIO)

16.1 Introduction . . . 405
16.2 Features . . . 405
16.3 Memory Map/Register Definition . . . 406
16.3.1 Register Descriptions . . . 407
16.4 Functional Description . . . 412

Chapter 17
Integrated Programmable Interrupt Controller (IPIC)

17.1 Introduction . . . 413
17.1.1 Overview . . . 416
17.1.2 Features . . . 416
17.1.3 Modes of Operation . . . 416
17.2 Memory Map/Register Definition . . . 417
17.2.1 Register Summary . . . 417
17.3 Functional Description . . . 447
17.3.1 Interrupt Types . . . 447
17.3.2 Interrupt Configuration . . . 447
17.3.3 Internal Interrupts Group Relative Priority . . . 450
17.3.4 Mixed Interrupts Group Relative Priority . . . 451
17.3.5 Highest Priority Interrupt . . . 451
17.3.6 Interrupt Source Priorities . . . 451
17.3.7 Masking Interrupt Sources . . . 455
17.3.8 Interrupt Vector Generation and Calculation . . . 456
17.3.9 Machine Check Interrupts . . . 456

Chapter 18
IIM/Fusebox

18.1 Introduction . . . 457
18.2 Overview . . . 457
18.2.1 Features . . . 457
18.2.2 Modes of Operation . . . 457
18.3 Memory Map and Register Definition . . . 457
18.3.1 Memory Map . . . 458
18.3.2 Register Descriptions . . . 458
18.4 Functional Description . . . 468

18.4.1 Fuse Bank 1 468

Chapter 19
Inter-Integrated Circuit (I^2C)

19.1 Overview 469
19.1.1 Features 470
19.1.2 I2C Controller 471
19.1.3 START Signal 471
19.1.4 STOP Signal 472
19.1.5 Acknowledge 473
19.1.6 Arbitration 475
19.2 External Signal Description 475
19.3 Memory Map and Register Definition 476
19.3.1 Register Descriptions 477
19.4 Initialization Sequence 490
19.5 Transfer Initiation and Interrupt 491
19.5.1 Post-Transfer Software Response 491
19.5.2 Slave Mode 492
19.5.3 Special Note on AKF 492

Chapter 20
I/O Control

20.1 Introduction 495
20.1.1 Overview 495
20.1.2 Features 495
20.2 Memory Map and Register Definition 495
20.2.1 Memory Map 495
20.2.2 Register Descriptions 502
20.3 Application Information 518

Chapter 21
LocalPlus Bus Controller (LPC)

21.1 Introduction 519
21.1.1 Features 519
21.2 Memory Map and Register Definition 521
21.2.1 Register Descriptions 522
21.3 Functional Description 540
21.3.1 Non-Muxed Mode 540
21.3.2 Muxed Mode 548
21.3.3 SCLPC Interface 555
21.3.4 Programmer’s Model 556

Chapter 22
MSCAN

22.1 Introduction . . . 557
22.1.1 Features . . . 557
22.2 External Signal Description . . . 558
22.2.1 CAN Receiver Input Pins . . . 559
22.2.2 CAN Transmitter Output Pins . . . 559
22.2.3 CAN System . . . 559
22.3 Memory Map and Register Definition . . . 560
22.3.1 Register Summary . . . 564
22.3.2 Register Descriptions . . . 567
22.3.3 Programmer's Model of Message Storage . . . 582
22.4 Functional Description . . . 592
22.4.1 General . . . 592
22.4.2 Message Storage . . . 592
22.4.3 Identifier Acceptance Filter . . . 595
22.4.4 Protocol Violation Protection . . . 599
22.4.5 Clock System . . . 599
22.4.6 Timer Link . . . 601
22.4.7 Modes of Operation . . . 602
22.4.8 Low Power Options . . . 602
22.4.9 Reset Initialization . . . 606
22.4.10 Interrupts . . . 606
22.4.11 Description of Interrupt Operation . . . 606
22.4.12 Interrupt Acknowledge . . . 607
22.4.13 Recovery from Deep Sleep Mode . . . 608
22.4.14 MSCAN Initialization . . . 608
22.4.15 Bus-Off Recovery . . . 609

Chapter 23
NAND Flash Controller (NFC)

23.1 Introduction . . . 611
23.2 Overview . . . 612
23.3 Features . . . 612
23.4 External Signal Description . . . 612
23.4.1 Overview . . . 612
23.5 NFC Buffer Memory Space . . . 613
23.6 Memory Map and Register Definition . . . 614
23.6.1 Memory Map . . . 614
23.6.2 Register Summary . . . 616
23.7 Register Descriptions . . . 618
23.7.1 Flash Command 1 register (FLASH_CMD1) . . . 618
23.7.2 Flash Command 2 register (FLASH_CMD2) . . . 618
23.7.3 Column Address register (COL_ADDR) . . . 619

23.7.4 Row Address register (ROW_ADDR) . . . 620
23.7.5 Flash Command Repeat register (FLASH_COMMAND_REPEAT) . . . 620
23.7.6 Row Address Increment register (ROW_ADDR_INC) . . . 621
23.7.7 Flash Status 1 register (FLASH_STATUS1) . . . 622
23.7.8 Flash Status 2 register (FLASH_STATUS2) . . . 622
23.7.9 DMA1 Address register (DMA1_ADDR) . . . 623
23.7.10 DMA Configuration register (DMA_CONFIG) . . . 623
23.7.11 Cache Swap register (CACHE_SWAP) . . . 624
23.7.12 Sector Size register (SECTOR_SIZE) . . . 625
23.7.13 Flash Configuration register (FLASH_CONFIG) . . . 625
23.7.14 DMA2 Address register (DMA2_ADDR) . . . 627
23.7.15 IRQ and Status register (IRQ_STATUS) . . . 628
23.8 Functional Description . . . 629
23.8.1 Error Corrector Status . . . 631
23.8.2 NFC Basic Commands . . . 631
23.8.3 NAND Flash boot . . . 637
23.8.4 Fast Flash Configuration for EDO . . . 639
23.8.5 Organization of Data in the NAND Flash . . . 639
23.8.6 Flash Command Sequencer . . . 642

Chapter 24
Power Management Control Module (PMC)

24.1 Introduction . . . 645
24.1.1 Features . . . 645
24.2 Memory Map and Register Definition . . . 645
24.2.1 Memory Map . . . 645
24.2.2 Register Descriptions . . . 646
24.3 Functional Description . . . 652
24.3.1 Full-Power Mode . . . 653
24.3.2 Doze Mode . . . 653
24.3.3 Nap Mode . . . 654
24.3.4 Sleep Mode . . . 654
24.3.5 Deep Sleep Mode . . . 655
24.3.6 Core PLL Change Mode . . . 656
24.3.7 PRE_DIV Copy Enable Mode . . . 656
24.3.8 Low-Power Configurations . . . 657

Chapter 25
Programmable Serial Controller (PSC)

25.1 Introduction . . . 659
25.2 Features . . . 660
25.3 PSC Functions Overview . . . 661
25.4 Memory Map and Registers . . . 662
25.4.1 Register Descriptions . . . 665

25.5 Modes of Operation 690
25.5.1 PSC in UART Mode 690
25.5.2 PSC in Codec Mode 697
25.5.3 PSC in AC97 Mode 707
25.5.4 Local Loop-Back Mode 711
25.5.5 Remote Loop-Back Mode 711
25.5.6 Multidrop Mode 712

Chapter 26
PSC Centralized FIFO Controller (FIFOC)

26.1 Introduction 715
26.1.1 Features 716
26.1.2 Modes of Operation 716
26.2 Memory Map and Register Definition 717
26.2.1 Register Descriptions 720
26.3 Functional Description 729

Chapter 27
Real Time Clock (RTC)

27.1 Introduction 731
27.1.1 Features 732
27.2 External Signal Descriptions 733
27.3 Memory Map and Register Definition 733
27.3.1 Memory Map 733
27.3.2 Register Descriptions 734
27.4 Functional Description 746
27.4.1 Behavior at Power On 746
27.4.2 Behavior of Wakeup Sources 746
27.4.3 Behavior During Power Off (Hibernation Mode) 747
27.4.4 RTC Response to Target Time Register/Actual Time Count Register and External Wakeup Sources 749
27.4.5 RTC Response to External Wakeup Sources 751

Chapter 28
Secure Digital Host Controller (SDHC)

28.1 Introduction 755
28.1.1 Features 755
28.2 External Signal Description 756
28.2.1 Detailed Signal Descriptions 756
28.3 Memory Map and Register Definition 757
28.3.1 Memory Map 757
28.3.2 Register Descriptions 758
28.4 Functional Description 777
28.4.1 Data Buffers 777

28.4.2 DMA Interface . . . 781
28.4.3 Memory Controller . . . 782
28.4.4 SDIO Card Interrupt . . . 783
28.4.5 Card Insertion and Removal Detection . . . 785
28.4.6 Power Management . . . 786
28.4.7 System Clock Controller . . . 786
28.5 Initialization Information . . . 787
28.5.1 MMC_SD_CLK Control . . . 788
28.5.2 Command Submit – Response Receive Basic Operation . . . 788
28.5.3 Card Identification Mode . . . 789
28.5.4 Card Access . . . 793
28.5.5 Switch Card Mode . . . 796

Chapter 29
Software Watchdog Timer (WDT)

29.1 Introduction . . . 799
29.1.1 Features . . . 799
29.1.2 Modes of Operation . . . 799
29.2 Memory Map/Register Definition . . . 800
29.2.1 Memory Map . . . 800
29.2.2 Register Descriptions . . . 800
29.3 Functional Description . . . 803
29.3.1 Software Watchdog Timer Unit . . . 803
29.3.2 Modes of Operation . . . 804

Chapter 30
SRAM Memory (MEM)

30.1 Introduction . . . 807

Chapter 31
Temperature Sensor

31.1 Introduction . . . 809
31.1.1 Normal Operation Mode . . . 809

Chapter 32
Universal Serial Bus Interface with On-The-Go

32.1 Introduction . . . 811
32.1.1 Overview . . . 811
32.1.2 Features . . . 811
32.1.3 Modes of Operation . . . 812
32.2 Memory Map/Register Definitions . . . 812
32.2.1 Module Identification Registers . . . 817
32.2.2 Capability Registers . . . 822
32.2.3 Device/Host Timer Registers (Non-EHCI) . . . 826

32.2.4 Operational Registers 828
32.3 Functional Description 864
32.3.1 System Interface 864
32.3.2 DMA Engine 865
32.3.3 FIFO RAM Controller 865
32.4 OTG Operations 865
32.4.1 Register Bits 865
32.4.2 Hardware Assist 866
32.5 Host Data Structures 868
32.5.1 Periodic Frame List 868
32.5.2 Asynchronous List Queue Head Pointer 870
32.5.3 Isochronous (High-Speed) Transfer Descriptor (iTD) 870
32.5.4 Split Transaction Isochronous Transfer Descriptor (siTD) 874
32.5.5 Queue Element Transfer Descriptor (qTD) 878
32.5.6 Queue Head 882
32.5.7 Periodic Frame Span Traversal Node (FSTN) 887
32.6 Host Operational Model 888
32.6.1 Host Controller Initialization 889
32.6.2 Suspend/Resume 890
32.6.3 Schedule Traversal Rules 892
32.6.4 Periodic Schedule Frame Boundaries vs. Bus Frame Boundaries 894
32.6.5 Periodic Schedule 896
32.6.6 Managing Isochronous Transfers Using iTDs 897
32.6.7 Asynchronous Schedule 901
32.6.8 Operational Model for NAK Counter 909
32.6.9 Managing Control/Bulk/Interrupt Transfers via Queue Heads 911
32.6.10 Ping Control 922
32.6.11 Split Transactions 923
32.6.12 Host Controller Pause 951
32.6.13 Port Test Modes 952
32.6.14 Interrupts 952
32.7 Device Data Structures 957
32.7.1 Endpoint Queue Head 957
32.7.2 Endpoint Transfer Descriptor (dTD) 960
32.8 Device Operational Model 962
32.8.1 Device Controller Initialization 962
32.8.2 Port State and Control 963
32.8.3 Bus Reset 966
32.8.4 Managing Endpoints 967
32.8.5 Device Operational Model For Packet Transfers 969
32.8.6 Managing Queue Heads 977
32.8.7 Managing Transfers with Transfer Descriptors 979
32.8.8 Device Error Matrix 981
32.8.9 Servicing Interrupts 982
32.8.10 Deviations from the EHCI Specifications 983

About this book

This reference manual describes the MPC5125 microcontroller family for software and hardware developers. Information regarding bus timing, signal behavior, and AC, DC, and thermal characteristics are detailed in the device data sheet (*MPC5125 Data Sheet*).

The information in this book is subject to change without notice, as described in the disclaimers on the title page. As with any technical documentation, the reader needs to make sure to use the most recent version of the documentation.

To locate any published errata or updates for this document, visit the Freescale web site at http://www.freescale.com/.

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the MPC5125 microcontroller family. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the Power architecture.

Organization

Following is a summary and brief description of the major sections of this manual:

- Chapter 1, “Overview,” includes general descriptions of the modules and features incorporated in the device while focusing on new features.
- Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map),” describes the memory map and configuration for the MPC5125.
- Chapter 3, “Signal Descriptions,” summarizes the external signal functions, their static electrical characteristics, and pad configuration settings for the MPC5125.
- Chapter 4, “Reset,” describes the reset sources available on the MPC5125, including details on status flags and default configurations.
- Chapter 5, “Clocks and Low-Power Modes,” describes the various clock sources that are available on the MPC5125 device.
- Chapter 6, “Byte Data Link Controller (BDLC),” describes the BDLC, which is a Society of Automotive Engineers (SAE) J1850-compatible serial network communication module.
- Chapter 7, “CPU e300 Core Power Architecture,” describes the organization of the Power processor core on the MPC5125, and gives an overview of the programming models as they are implemented on the device.
- Chapter 8, “CSB Arbiter and Bus Monitor,” describes the Coherent Systems Bus (CSB) arbiter and its configuration, control, and status registers.

- Chapter 9, “Direct Memory Access (DMA),” describes the DMA controller implemented on the MPC5125.
- Chapter 10, “Display Interface Unit (DIU),” describes the Display Interface Unit (DIU) implemented on the MPC5125.
- Chapter 11, “DRAM Controller,” describes the multi-port DRAM controller that supports Mobile-DDR, DDR-1, DDR-2, and SDR memories.
- Chapter 12, “Multi-port DRAM Controller Priority Manager,” describes the priority manager for the DRAM controller.
- Chapter 13, “External Memory Bus (EMB),” describes how the LPC and the NFC share the External Memory Bus.
- Chapter 14, “Fast Ethernet Controller (FEC),” describes the feature set, operation, and programming model of the FEC block.
- Chapter 15, “General Purpose Timers (GPT),” describes eight independent timer channels that perform general purpose timer and general purpose I/O (GPIO) functions.
- Chapter 16, “General Purpose I/O (GPIO),” describes the general purpose I/O module, including pin descriptions, register settings and interrupt capabilities.
- Chapter 17, “Integrated Programmable Interrupt Controller (IPIC),” summarizes the software and hardware interrupts for the MPC5125 device.
- Chapter 18, “IIM/Fusebox,” describes the module that provides an interface for reading and programming information stored in on-chip fuse elements.
- Chapter 19, “Inter-Integrated Circuit (I2C),” describes the I^2C module, including I^2C protocol, clock synchronization, and I^2C programming model registers.
- Chapter 20, “I/O Control,” describes the controls for the functional muxing and configuration of the pads.
- Chapter 21, “LocalPlus Bus Controller (LPC),” describes the external bus interface of the MPC5125.
- Chapter 22, “MSCAN,” describes the CAN module, a communication controller implementing the CAN protocol according to Bosch Specification version 2.0B.
- Chapter 23, “NAND Flash Controller (NFC),” describes the interface to standard NAND flash memory devices.
- Chapter 24, “Power Management Control Module (PMC),” describes the power blocks that provide voltage control for the internal logic of the device.
- Chapter 25, “Programmable Serial Controller (PSC),” describes the UART, AC97, Codec, and I^2S modes.
- Chapter 26, “PSC Centralized FIFO Controller (FIFOC),” describes the centralized FIFO controller for the PSC modules.
- Chapter 27, “Real Time Clock (RTC),” describes the real time clock.
- Chapter 28, “Secure Digital Host Controller (SDHC),” describes the module that interfaces to Multimedia Cards (MMC), Secure Digital (SD) memory cards, and I/O cards.
- Chapter 29, “Software Watchdog Timer (WDT),” describes the counter that guards against software errors by periodically issuing a reset unless interrupted by software.

- Chapter 30, “SRAM Memory (MEM),” describes the on-chip static RAM (SRAM) implementation.
- Chapter 31, “Temperature Sensor,” describes the module that monitors the internal temperature of the MPC5125.
- Chapter 32, “Universal Serial Bus Interface with On-The-Go,” describes the universal serial bus (USB) interface on the MPC5125.
- Appendix A, “Memory Map,” provides a detailed listing of the memory-mapped registers for the MPC5125.

Suggested reading

This section lists additional reading that provides background for the information in this manual as well as general information about PowerPC architecture.

General information

Useful information about the PowerPC architecture and computer architecture in general:

- *Programming Environments Manual for 32-Bit Implementations of the PowerPC™ Architecture* (MPCFPE32B)
- *Using Microprocessors and Microcomputers: The Motorola Family*, William C. Wray, Ross Bannatyne, Joseph D. Greenfield
- *Computer Architecture: A Quantitative Approach*, Second Edition, by John L. Hennessy and David A. Patterson.
- *Computer Organization and Design: The Hardware/Software Interface*, Second Edition, David A. Patterson and John L. Hennessy.

PowerPC documentation

PowerPC documentation is available from the sources listed on the back cover of this manual, as well as http://www.freescale.com/powerarchitecture .

- Reference manuals (formerly called user’s manuals)—These books provide details about individual PowerPC implementations and are intended to be used in conjunction with the *PowerPC Programmers Reference Manual.*
- Addenda/errata to reference manuals—Because some processors have follow-on parts, an addendum is provided that describes the additional features and functionality changes. Also, if mistakes are found within a reference manual, an errata document will be issued before the next published release of the reference manual. These addenda/errata are intended for use with the corresponding reference manuals.
- Data sheets—Data sheets provide specific information regarding pin-out diagrams, bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations.
- Product briefs—Each device has a product brief that provides an overview of its features. This document is roughly equivalent to the overview (Chapter 1) of a device’s reference manual.

- Application notes—These short documents address specific design issues useful to programmers and engineers working with Freescale Semiconductor processors.

Additional literature is published as new processors become available. For a current list of PowerPC documentation, refer to http://www.freescale.com/powerarchitecture .

Conventions

This document uses the following notational conventions:

cleared/set	When a bit takes the value zero, it is said to be cleared; when it takes a value of one, it is said to be set.
reserved	When a bit or address is reserved, it should not be written. If read, its value is cannot be not guaranteed. Reading or writing to reserved bits or addresses may cause unexpected results.
MNEMONICS	In text, instruction mnemonics are shown in uppercase.
`mnemonics`	In code and tables, instruction mnemonics are shown in lowercase.
italics	Italics indicate variable command parameters. Book titles in text are set in italics.
0b0	Prefix to denote binary number (e.g., 0b0110_0111)
0x0	Prefix to denote hexadecimal number (e.g., 0xFFF0_FFFC)
b	Suffix to denote binary number (e.g., 0110_0111b)
d	Suffix to denote decimal number (e.g., 12345d)
h	Suffix to denote hexadecimal number (e.g., FFF0_FFFCh)
REG[FIELD]	Abbreviations for registers are shown in uppercase. Specific bits, fields, or ranges appear in brackets. For example, RAMBAR[BA] identifies the base address field in the RAM base address register.
nibble	A 4-bit data unit
byte	An 8-bit data unit
halfword	A 16-bit data unit[1]
word	A 32-bit data unit
doubleword	A 64-bit data unit
x	In some contexts, such as signal encodings, x (without italics) indicates a “don’t care” condition.
x	With italics, used to express an undefined alphanumeric value (e.g., a variable in an equation); or a variable alphabetic character in a bit, register, or module name (e.g., DSPI_*x* could refer to DSPI_A or DSPI_B).
n	Used to express an undefined numerical value; or a variable numeric character in a bit, register, or module name (e.g., EIF*n* could refer to EIF1 or EIF0).

1. The only exceptions to this appear in the discussion of serial communication modules that support variable-length data transmission units. To simplify the discussion these units are referred to as words regardless of length.

~	NOT logical operator
&	AND logical operator
|	OR logical operator
||	Field concatenation operator
$\overline{\text{OVERBAR}}$	An overbar indicates that a signal is active-low.

Register figure conventions

This document uses the following conventions for the register reset values in register figures:

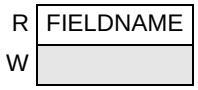
—	Bit value is undefined at reset.
U	Bit value is unchanged by reset. Previous value preserved during reset.
[*signal_name*]	Reset value is determined by the polarity of the indicated signal.

The following descriptions are used in register bit field description tables:

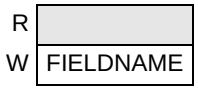
Indicates a reserved bit field in a memory-mapped register. These bits are always read as 0.

Indicates a reserved bit field in a memory-mapped register. These bits are always read as 1.

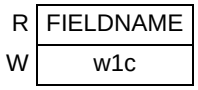
Indicates a read/write bit in a memory-mapped register.



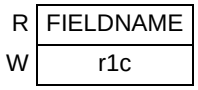
Indicates a read-only bit field in a memory-mapped register.



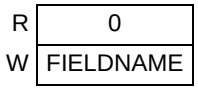
Indicates a write-only bit field in a memory-mapped register.



Write 1 to clear: indicates that writing a 1 to this bit field clears it.



Read to clear: indicates that reading this bit field clears it, regardless of its returned value.



Indicates a self-clearing bit.

Acronyms and Abbreviations

Table i lists some acronyms and abbreviations used in this document.

Table i. Acronyms and Abbreviated Terms

Term	Meaning
ADC	Analog-to-digital converter
ALU	Arithmetic logic unit
BDM	Background debug mode
BIST	Built-in self test
BSDL	Boundary-scan description language
CODEC	Code/decode
DAC	Digital-to-analog conversion
DMA	Direct memory access
DSP	Digital signal processing
EA	Effective address
FIFO	First-in, first-out
GPIO	General-purpose I/O
I^2C	Inter-integrated circuit
IEEE	Institute for Electrical and Electronics Engineers
IFP	Instruction fetch pipeline
IPL	Interrupt priority level
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
LIFO	Last-in, first-out
LRU	Least recently used
LSB	Least-significant byte
LVI	Low-voltage interrupt
lsb	Least-significant bit
MAC	Multiply accumulate unit, also Media access controller
MSB	Most-significant byte
msb	Most-significant bit
Mux	Multiplex
NOP	No operation
OEP	Operand execution pipeline
PC	Program counter

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
PLIC	Physical layer interface controller
PLL	Phase-locked loop
POR	Power-on reset
RISC	Reduced instruction set computing
Rx	Receive
SOF	Start of frame
TAP	Test access port
TTL	Transistor transistor logic
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter
USB	Universal serial bus

Terminology Conventions

Table ii shows terminology conventions used throughout this document.

Table ii. Notational Conventions

Instruction	Operand Syntax
Opcode Wildcard	
cc	Logical condition (example: NE for not equal)
Register Specifications	
An	Any address register n (example: A3 is address register 3)
Ay,Ax	Source and destination address registers, respectively
Dn	Any data register n (example: D5 is data register 5)
Dy,Dx	Source and destination data registers, respectively
Rc	Any control register (example VBR is the vector base register)
Rm	MAC registers (ACC, MAC, MASK)
Rn	Any address or data register
Rw	Destination register w (used for MAC instructions only)
Ry,Rx	Any source and destination registers, respectively
Xi	Index register i (can be an address or data register: Ai, Di)
Miscellaneous Operands	
#<data>	Immediate data following the 16-bit operation word of the instruction
<ea>	Effective address
<ea>y,<ea>x	Source and destination effective addresses, respectively
<label>	Assembly language program label
<list>	List of registers for MOVEM instruction (example: D3–D0)
<shift>	Shift operation: shift left (<<), shift right (>>)
<size>	Operand data size: byte (B), word (W), longword (L)
bc	Instruction and data caches
dc	Data cache
ic	Instruction cache
# <vector>	Identifies the 4-bit vector number for trap instructions
<>	identifies an indirect data address referencing memory
<xxx>	identifies an absolute address referencing memory
d*n*	Signal displacement value, *n* bits wide (example: d16 is a 16-bit displacement)
SF	Scale factor (x1, x2, x4 for indexed addressing mode, <<1n>> for MAC operations)

Table ii. Notational Conventions (continued)

Instruction	Operand Syntax
Operations	
+	Arithmetic addition or postincrement indicator
–	Arithmetic subtraction or predecrement indicator
x	Arithmetic multiplication
/	Arithmetic division
~	Invert; operand is logically complemented
&	Logical AND
|	Logical OR
^	Logical exclusive OR
<<	Shift left (example: D0 << 3 is shift D0 left 3 bits)
>>	Shift right (example: D0 >> 3 is shift D0 right 3 bits)
→	Source operand is moved to destination operand
←→	Two operands are exchanged
sign-extended	All bits of the upper portion are made equal to the high-order bit of the lower portion
If <condition> then <operations> else <operations>	Test the condition. If true, the operations after then are performed. If the condition is false and the optional else clause is present, the operations after else are performed. If the condition is false and else is omitted, the instruction performs no operation. Refer to the Bcc instruction description as an example.
Subfields and Qualifiers	
{}	Optional operation
()	Identifies an indirect address
d_n	Displacement value, n-bits wide (example: d_{16} is a 16-bit displacement)
Address	Calculated effective address (pointer)
Bit	Bit selection (example: Bit 3 of D0)
lsb	Least significant bit (example: lsb of D0)
LSB	Least significant byte
LSW	Least significant word
msb	Most significant bit
MSB	Most significant byte
MSW	Most significant word

Revision History

Table iii provides a revision history for this document.

Table iii. MPC5125 Microcontroller Reference Manual Revision History

Revision Number	Revision Date	Description of Changes
1	06/2008	Release 1 of this document.
2	7/2009	Release 2 of this document. Significant technical and editorial revision.

Table iii. MPC5125 Microcontroller Reference Manual Revision History (continued)

Revision Number	Revision Date	Description of Changes
3	10/2011	—Published for review as MPC5125RM Rev. 3.
	Chapter 2	—Updated Note in Section “ LocalPlus Boot Access Window Register (LPBAW)” —Updated Table” LPBAW/LPCSnAW [START_ADDR] and [STOP_ADDR] Reset Values” —Updated Table”NFCBAR[BASE_ADDR] Reset Value” —Updated Table “Memory Map Values based on EMBAD0/1 and BMS” —Updated Figure”Initial Memory Map Configurations Immediately After the Release of PORESET”
	Chapter 3	—Updated figure “MPC5125 pin signal groupings”. FEC1_TX_ER (arrow pointing into MPU) changed to FEC1_TX_ER (arrow pointing away from MPU);FEC1_MDC (arrow pointing into MPU) changed to —Updated table “MPC5125 pin multiplexing”. FEC1_TX_ER (I/O Direction: I) changed to FEC1_TX_ER (I/O Direction: O);FEC1_MDC (I/O Direction: I) changed to FEC1_MDC (I/O Direction: O);FEC2_TX_ER (I/O Direction: I) change to FEC2_TX_ER (I/O Direction: O);FEC2_MDC (I/O Direction: I) changed to FEC2_MDC (I/O Direction: O) —Updated table “MPC5125 pin multiplexing”. FEC1_TX_CLK (I/O Direction: O) changed to FEC1_TX_CLK(I/O Direction: I). —Updated table “MPC5125 pin multiplexing”. NFC_R/B changed to NFC_R/B0 for ALT0 of NFC_RB ; the ALT2 function of the PSC1_3 signal lists NFC_R/B2 signal direction changed as an input; the ALT2 function of the J1850_RX signal lists NFC_R/B3 signal direction changed as an input. —Updated figure “MPC5125 pin signal groupings” ,NFC_R/B changed to NFC_R/B0 and FEC signal updated. —Updated table “ Pin Multiplexing”. "ALT3" replaced with "ALT2" for "RST_CONF" (reset configuration);FEC1_MDIO/RMII_MDIO I/O direction changed from I to I/O ; FEC_TX_EN I/O direction changed to O from I;USB1_DATA1and USB1_NEXT I/O direction changed to O from I.
	Chapter 8	—Updated Table “ACR field descriptions”.
	Chapter 14	—Updated figure "FEC block diagram". FECn_TX_ER (arrow pointing into MPU) changed to FECn_TX_ER (arrow pointing away from MPU). —Updated figure " FEC block diagram", FEC1_TX_CLK (I/O Direction: O) changed to FEC1_TX_CLK(I/O Direction: I). —Updated figure " FEC block diagram", all MII/RMII/7-wire signal arrow direction reversed. —Updated section" Ethernet Address Recognition" in Chapter "Fast Ethernet Controller".
	Chapter 20	—Updated Table “I/O Control Memory Map” in Section “Memory Map”.Updated Register value of row 0x04B and 0x04C.
	Chapter 23	—Updated figure "Flash Command 2 register (FLASH_CMD2)".The location of BUFNO bits changed to 29:30 from 30:31 with the BUSY/START bit as bit 31. —Updated table “FLASH_CMD2 field descriptions”. —Updated table “SECTOR_SIZE field descriptions”.
	Chapter 25	—Updated figure “Auxiliary Control Register for all Modes (ACR)”, IEC1 and IEC0 bit fields added to the register figure.
	Chapter 29	—Updated table "WDT memory map", 0x0004-0xFF changed to 0x0010-0xFF —Updated table WDT memory map, 0x0010-0xFF changed to 0x10-0xFF.

Chapter 1
Overview

1.1 Introduction

The MPC5125 integrated processor provides a value-based computing platform for telematics vehicle applications for OEM, aftermarket, and commercial products. The MPC5125 is also excellent for any embedded solution that requires network connectivity, display controller, low power consumption and an extensive I/O set. The MPC5125 has automotive qualification; therefore, all customers can expect competitive cost, quality, reliability and availability for years to come. The MPC5125 uses the e300 CPU core based on the Power Architecture™ instruction set.

The MPC5125 integrated processor includes one core and multiple buses, helping to avoid bandwidth bottlenecks. The excellent balance between operating power consumption and performance allows for lower system cost and higher reliability. The low standby power consumption also makes the product suitable for portable applications.

The flexibility of the MPC5125 provides customers a platform for a variety of product applications. Its rich set of integrated I/O includes Ethernet, USB 2.0, CAN, 10 programmable serial controllers, and numerous others. The integrated display controller (DIU) allows for cost-effective support of thin film transistor (TFT) LCD panel displays.

The MPC5125 uses the e300 CPU core, with 32 KB instruction cache and 32 KB data cache, based on the Power Architecture™ instruction set. Wide support of Linux, RTOS, software drivers, middleware, and application solutions from mobileGT alliance members is planned. This can greatly reduce development lead times and expense while improving software quality. The MPC5125 is designed for application code compatibility with current solutions on the MPC5121e/MPC5200 and for easy scalability across product solutions.

The many embedded memory buffers help ensure balanced system performance and system bus throughput. The performance of the MPC5125 is enhanced by having well-balanced system resources for the integrated core, graphics and audio engines, DDR1/DDR2/SDR/Mobile DDR memory controllers and integrated 64-channel DMA support.

Figure 1-1 shows a top-level block diagram of the MPC5125.

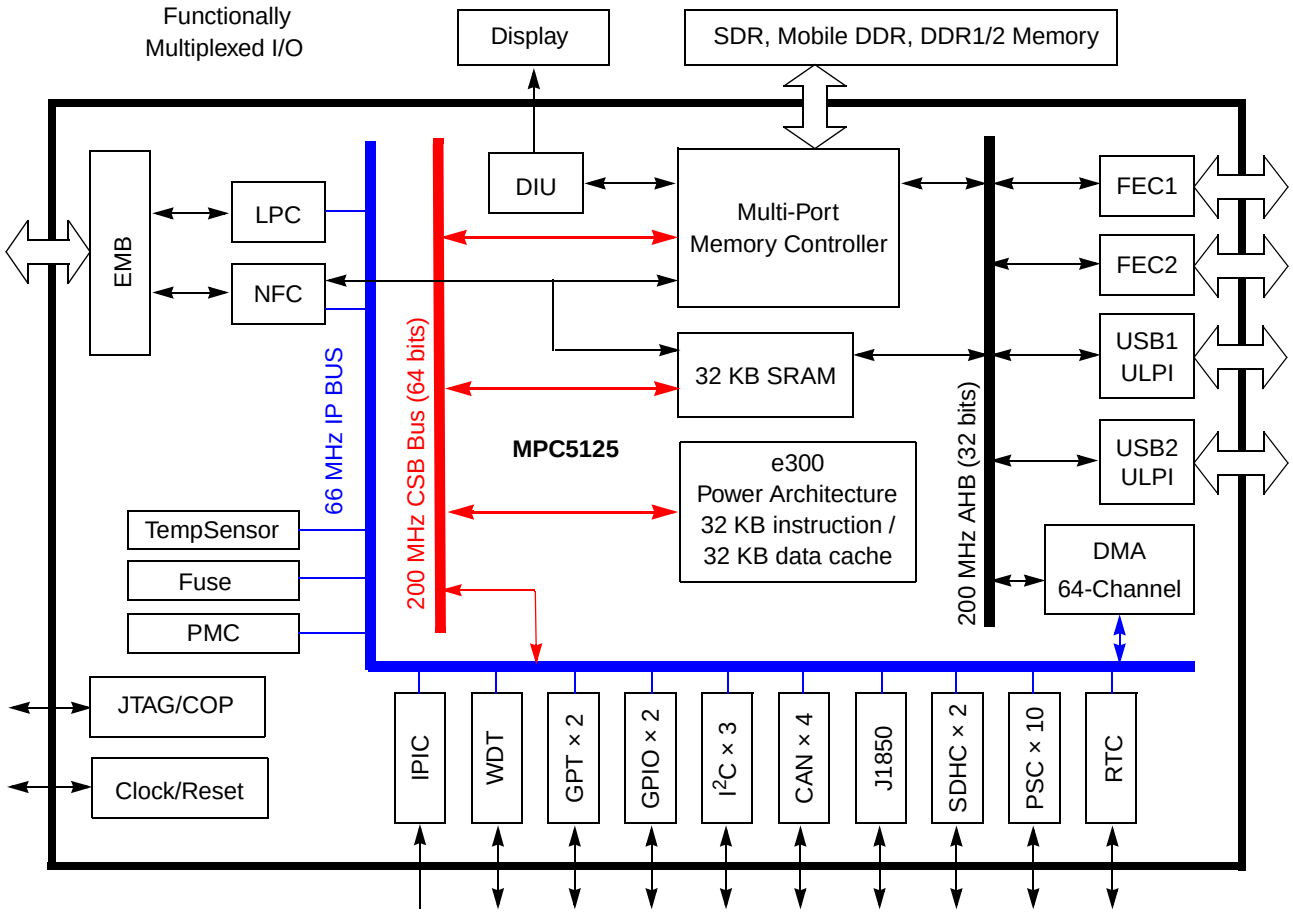


Figure 1-1. MPC5125 Block Diagram

1.2 Chip-Level Features

Major features of the MPC5125 are as follows:

- e300 Power Architecture processor core (enhanced version of the MPC603e core), operates as fast as 400 MHz
- Low power design
- Display interface unit (DIU)
- DDR1, DDR2, low-power mobile DDR (LPDDR), and 1.8 V/3.3 V SDR DRAM memory controllers
- 32 KB on-chip SRAM
- USB 2.0 OTG controller with ULPI interface
- DMA subsystem
- Flexible multi-function external memory bus (EMB) interface
- NAND flash controller (NFC)
- LocalPlus interface (LPC)
- 10/100Base Ethernet

- MMC/SD/SDIO card host controller (SDHC)
- Programmable serial controller (PSC)
- Inter-integrated circuit (I^2C) communication interfaces
- Controller area network (CAN)
- J1850 byte data link controller (BDLC) interface
- On-chip real-time clock (RTC)
- On-chip temperature sensor
- IC Identification module (IIM)

1.3 Module Features

The following provides more details of modules implemented on the device:

1.3.1 e300 Processor Core

- Power Architecture instruction set
- 32 KB instruction cache
- 32 KB data cache
- High-performance, superscalar processor core with a four-stage pipeline
- Dual-issue processor with integrated floating-point unit and dual integer units
- Dynamic power management

1.3.2 Display Interface Unit (DIU)

- Supports LCD display resolution as high as 1280 × 720
- Color depth as high as 24 bits per pixel
- Hardware n-plane (n ≥ 3) alpha blending, chroma keying support
- 32 × 32, 4,096 color hardware cursor

1.3.3 USB Controller

- Two on-chip USB controllers with On-The-Go (OTG) host/device capability
- Each supports high-speed (480 Mbps), full-speed (12 Mbps), and low-speed (1.5 Mbps)
- Both USB controllers can be accessed through the ULPI interface

1.3.4 Direct Memory Access (DMA) Controller

- 64-channel on-chip DMA engine with advanced capabilities
- Supports channel linking and scatter/gather processing

1.3.5 DDR SDRAM Memory Controller

- Supports 16-bit wide and 32-bit wide DDR1, DDR2, and LPDDR, 32-bit SDR SDRAM devices at speeds as fast as 200 MHz
- Supports two chip selects

1.3.6 32 KB On-chip SRAM

- Usable as e300 core scratch pad memory

1.3.7 Fast Ethernet Controller (FEC)

- Two Ethernet controllers
- Supports 100 Mbps/10 Mbps IEEE 802.3 MII
- Supports 10 Mbps 7-wire interface
- IEEE 802.3 full duplex flow control
- Supports RMII interface

1.3.8 NAND Flash Interface

- Supports NAND flashes with 8-bit or 16-bit data width
- Supports booting from page size ≥ 2 KB NAND devices
- Supports 512 byte/2 KB/4 KB/8 KB page NAND devices
- Supports four chip selects
- BCH ECC engine, 0/4/6/8/12/16/24/32-bit error correction mode

1.3.9 Local Plus Bus (LPC) Interface

- Interface to external memory-mapped or chip-selected devices
- 32-bit address bus
- 32-bit data bus
- Eight chip selects
- Supports burst mode flash
- Supports 32-bit ALE-muxed interface
- Supports as many as 42-bit non-muxed interfaces
- Supports large-packet DMA transfers

1.3.10 Secure Digital Host Controller (SDHC)

- Two SD/SDIO/MMC card interfaces
- Compliant with SD and SDIO specification version 1.x
- Compliant with MMC card specification

- 200 Mbps data rate in 4-bit mode

1.3.11 Controller Area Network (CAN)

- Four CAN interfaces
- Implementation of CAN protocol, version 2.0 A/B
- Programmable wakeup functionality
- Both support either low speed or high speed

1.3.12 Integrated Circuit Communication (I^2C)

- Three inter-integrated circuit (I^2C) communication interfaces
- Input digital noise filtering
- Master and slave modes supported

1.3.13 Programmable Serial Controller (PSC)

- Ten programmable serial controllers (PSC)
- Each PSC is a flexible serial communication engine, supporting the following protocols:
  - Universal asynchronous receiver/transmitter (UART)
  - Codec/Pulse-code modulation (PCM)
  - Inter-IC sound bus (I^2S)
  - Serial Peripheral Interface (SPI)
  - AC97
- Each PSC is serviced by a centralized FIFO to provide buffer storage

1.3.14 J1850 Byte Data Link Controller (BDLC) Interface

- SAE J1850 Class B data communications network interface compliant
- ISO-compatible for low speed (< 125 Kbps) serial data communications
- Digital noise filter

1.3.15 General Purpose I/Os (GPIO)

- Two general-purpose I/O (GPIO) modules, each with 32 pins
- Four GPIO pins in V_{BAT} power domain available for external wake up (GPIO1)

1.3.16 On-chip Real-time Clock (RTC)

- Real-time clock runs from separate V_{BAT} power domain
- Programmable alarm
- Periodic interrupts for one second, one minute, one day

- Runs with external 32 kHz crystal or external clock source

1.3.17 IC Identification Module (IIM)

- One fuse bank user space (32 bytes)

1.3.18 On-chip Temperature Sensor

- Capable of generating an interrupt at a programmable temperature level

1.3.19 Power Modes

- Run
- Doze
- Nap
- Sleep
- Deep sleep mode
- Hibernation mode

1.3.20 System Timers

- Real-time clock
- Two general-purpose timer (GPT) modules, each with eight separate timer channels

1.3.21 IEEE 1149.1 Compliant JTAG Boundary Scan

1.4 Developer Environment

The MPC5125 supports similar tools and third party developers as other mobileGT products, offering a widespread, established network of tools and software vendors.

The following development support is available.

- Evaluation/development boards
- IDE/tool chains
- C/C++ compilers
- Hardware and software debuggers
- Initialization/boot code generators
- Software libraries
- Device/module drivers
- JTAG interfaces
- Third party real-time operating systems (RTOS)

Chapter 2
System Configuration and Memory Map (XLBMEN + Mem Map)

2.1 Introduction

The System Configuration and Memory Map chapter describes the memory map of the MPC5125 and also details information on setting up the system configuration. The memory map is described by 13 local access windows. Some of these memory access windows point to various blocks of the 32-bit memory address space while other memory access windows point to specific peripheral modules. A list of the local access windows is shown in Table 2-1. The registers that set the base address and size of each local access window are presented in Table 2-2.

2.2 Memory Map and Register Definition

2.2.1 Local Memory Map Overview and Example

The MPC5125 provides a flexible local memory map. The local memory map refers to the 32-bit address space seen by the processor as it accesses memory and I/O space. Internal DMA engines also see this same local memory map. All memory accessed by the MPC5125 DDR SDRAM and LocalPlus controllers exists in this memory map, as do all memory-mapped configuration, control, and status registers.

The local memory map of the MPC5125 is defined by a set of 13 local access windows. Each of these windows maps a region of memory to a particular target interface, such as the DDR SDRAM controller. The LPC windows do not perform any address translation. Each local access window is assigned to a specific target interface as specified in Table 2-1.

Local Access Windows are defined in several different ways. For instance, Local Access Window 0 is used for the configuration registers which include the IMMRBAR register and the base address for registers in the various peripheral modules. The block size for Local Access Window 0 is fixed at 1 MB. Local Access Windows 1–9 are specified by the Start_Addr and Stop_Addr fields of the respective LocalPlus Access Window Registers for Chip Select Boot and Chip Selects 0–7. The DDR SDRAM window base address and size is specified by the DDR Local Access Window Base Address Register and the size of the window is specified by the DDR Local Access Window Attributes Register. Window 11 is used for the SRAM module. The SRAM module base address register is located at IMMRBAR + 0xC4. The block size is fixed at 32 KB; however, a 256 KB memory window is reserved for SRAM. Window 12 is used for the NAND Flash Controller. The NAND Flash Controller base address register is located at IMMR + 0xC8. The block size is fixed at 1 MB.

NOTE

It is generally a programming error to overlap the addressing range of the Local Access Windows. There is nothing to prevent software from programming overlapping addresses.

Changing the value of IMMRBAR changes the location of the Local Access Windows Target Interface. The address of the IMMRBAR Register is the contents of the IMMRBAR register, itself. A Special Purpose Register, SPR311, is specifically provided such that a copy of IMMRBAR can be maintained in a register that does not move in the memory map.

NOTE

It is the responsibility of the system software to properly maintain e300 special purpose register 311 (SPR311 -MBAR) such that its contents are the same as the contents of the IMMRBAR Register.

2.2.2 Address Translation and Mapping

Table 2-1. Local Access Windows Target Interface

Window Number	Offset to IMMR	Target Interface	Comments
0	0x000	Configuration registers (IMMRBAR)	Fixed 1 MB window size
1	0x020	LocalPlus Bus Boot Access Window Register	—
2	0x024	LocalPlus Bus CS0 Access Window Register	—
3	0x028	LocalPlus Bus CS1 Access Window Register	—
4	0x02C	LocalPlus Bus CS2 Access Window Register	—
5	0x030	LocalPlus Bus CS3 Access Window Register	—
6	0x034	LocalPlus Bus CS4 Access Window Register	—
7	0x038	LocalPlus Bus CS5 Access Window Register	—
8	0x03C	LocalPlus Bus CS6 Access Window Register	—
9	0x040	LocalPlus Bus CS7 Access Window Register	—
10	0x0A0	DDR SDRAM Local Access Window Base Address Register	—
	0x0A4	DDR SDRAM Local Access Window Attributes Register	—
11	0x0C4	SRAM Base Address Register	Fixed 256 KB window size
12	0x0C8	NFC Base Address Register	Fixed 1 MB window size

The local access windows perform target mapping for transactions within the local address space. No address translation is performed by the local access windows.

2.2.3 Window into Configuration Space

The internal memory map base address register (IMMRBAR) defines a window that is used to access all memory-mapped configuration, control, and status registers, referred as internal memory map registers or IMMR. The window is always enabled with a fixed size of 1 MB. There is no attributes register associated with Window 0. This window always takes precedence over all local access windows. The IMMRBAR

always comes out of reset with a default base address value of 0xFF40_0000. The value of IMMRBAR can be modified by writing to this register. For more information, see Section 2.2.5.1.1, “Internal Memory Map Registers Base Address Register (IMMRBAR).”

NOTE

Even though Window 0 uses only a 1 MB addressing range, it is recommended that it be treated as a 4 MB addressing range. For example, if IMMRBAR is set to 0xFF40_0000, reserve an address space of 0xFF40_0000–0xFF7F_FFFF. It is legal to use the 3 MB address space directly above Window 0 in the MPC5125. However, this space may be used in future derivatives of the MPC512x family of devices. Therefore, to maintain code compatibility between the MPC5125 and future derivatives of this family, it is recommended that this space not be used for user program code.

2.2.4 Local Access Windows

As demonstrated in the address map overview in Section 2.2.1, “Local Memory Map Overview and Example,” local access windows associate a range of the local 32-bit address space with a particular target interface. This allows the internal interconnections of the MPC5125 to route a transaction from its source to the proper target. No address translation is performed. The base address defines the high order address bits that give the location of the window in the local address space. The window attributes enable the window and define its size, while the window number specifies the target interface.

With the exception of configuration space (mapped by IMMRBAR), all addresses used by the system must be mapped by a local access window.

The local access window registers exist as part of the local access block in the system configuration registers. See Section 2.3.1.1, “System Configuration Registers.” A detailed description of the local access window registers is given in the following sections.

2.2.5 Local Access Register Memory Map

Table 2-2 shows the memory map for the local access registers.

Table 2-2. Local Access Register Memory Map

Offset from LOCAL_BASE (0xFF40_0000)	Register	Access[1]	Reset Value	Section/Page
0x000	Internal Memory Map Base Address Register (IMMRBAR) WINDOW 0 Base address set by the BASE_ADDR field of the IMMRBAR register. Size fixed at 1 MB. See Table 2-4 for individual module base addresses.	R/W	0xFF80_0000	2.2.5.1.1/2-11
0x004–0x01F	Reserved			

Table 2-2. Local Access Register Memory Map (continued)

Offset from LOCAL_BASE (0xFF40_0000)	Register	Access[1]	Reset Value	Section/Page
0x020	LocalPlus Boot Access Window register (LPBAW) WINDOW 1 Base address and size set by the START_ADDR and STOP_ADDR fields of the LPBAW register.	R/W	U[2]	2.2.5.1.3/2-15
0x024	LocalPlus CS0 Access Window register (LPCS0AW) WINDOW 2 Base address and size set by the START_ADDR and STOP_ADDR fields of the LPCS0AW register.	R/W		2.2.5.1.3/2-15
0x028	LocalPlus CS1 Access Window register (LPCS1AW) WINDOW 3 Base address and size set by the START_ADDR and STOP_ADDR fields of the LPCS1AW register.	R/W		2.2.5.1.3/2-15
0x02C	LocalPlus CS2 Access Window register (LPCS2AW) WINDOW 4 Base address and size set by the START_ADDR and STOP_ADDR fields of the LPCS2AW register.	R/W		2.2.5.1.3/2-15
0x030	LocalPlus CS3 Access Window register (LPCS3AW) WINDOW 5 Base address and size set by the START_ADDR and STOP_ADDR fields of the LPCS3AW register.	R/W		2.2.5.1.3/2-15
0x034	LocalPlus CS4 Access Window register (LPCS4AW) WINDOW 6 Base address and size set by the START_ADDR and STOP_ADDR fields of the LPCS4AW register.	R/W		2.2.5.1.3/2-15
0x038	LocalPlus CS5 Access Window register (LPCS5AW) WINDOW 7 Base address and size set by the START_ADDR and STOP_ADDR fields of the LPCS5AW register.	R/W		2.2.5.1.3/2-15
0x03C	LocalPlus CS6 Access Window register (LPCS6AW) WINDOW 8 Base Address and Size set by the START_ADDR and STOP_ADDR fields of the LPCS6AW register.	R/W		2.2.5.1.3/2-15
0x040	LocalPlus CS7 Access Window register (LPCS7AW) WINDOW 9 Base address and size set by the START_ADDR and STOP_ADDR fields of the LPCS7AW register.	R/W		2.2.5.1.3/2-15

Table 2-2. Local Access Register Memory Map (continued)

Offset from LOCAL_BASE (0xFF40_0000)	Register	Access[1]	Reset Value	Section/Page
0x044–0x09F	Reserved			
0x0A0	DDR Local Access Window0 Base Address register (DDRLAWBAR0) WINDOW 10 Base address set by the BASE_ADDR field of the DDRLAWBAR register.	R/W		2.2.5.1.5/2-17
0x0A4	DDR Local Access Window0 Attribute register (DDRLAWAR0) WINDOW 10 Size set by SIZE field of the DDRLAWAR register.	R/W		2.2.5.1.6/2-17
0x0A8–0x0C3	Reserved			
0x0C4	SRAM Address Register (SRAMBAR) WINDOW 11 Base address set by the BASE_ADDR field of the SRAMBAR register. Size fixed at 256 KB.	R/W		2.2.5.1.7/2-18
0x0C8	NFC Address Register (NFCBAR) WINDOW 12 Base address set by the BASE_ADDR field of the NFCBAR register. Size fixed at 1 MB.	R/W		2.2.5.1.8/2-18
0x0CC–0x0FF	Reserved			

1 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

2 The LPBAW reset value depends on the reset configuration word high values. See Table 2-6.

2.2.5.1 Local Access Register Description

2.2.5.1.1 Internal Memory Map Registers Base Address Register (IMMRBAR)

The internal memory map registers contain configuration, control, and status registers as well as device internal memory arrays. The internal memory map occupies a 1 MB region of memory space. Its location is programmable using the internal memory map register (IMMRBAR). The default base address for the internal memory map register is 0xFF40_0000. Because IMMRBAR is at offset 0x000 from the beginning of the local access registers, IMMRBAR always points to itself. A complete list of the modules that are memory mapped by the IMMRBAR is shown in Table 2-4.

2.2.5.1.2 Updating IMMRBAR

Updates to IMMRBAR relocate the entire 1 MB internal memory block. Thus, special considerations are required. The effect of the update must be guaranteed to be visible by the mapping logic before an access to the new location is accessed. To make sure this happens, these guidelines should be followed:

- IMMRBAR should be updated during initial configuration of the device when only one host or controller has access to the device
- During system initialization, immediately after the release of reset, system software should set IMMRBAR to the desired final location before enabling other I/O devices, which can become potential bus masters. A copy of the IMMRBAR value should be written to Special Purpose Register SPR311 (MBAR). Updating SPR311 is not automatic. When software changes IMMRBAR, SPR311 should be updated by software at the same time. Otherwise, addresses of configuration and data registers will be lost.

The Internal Memory Map Registers' Base Address register is shown in Figure 2-1.

Address: Base + 0x000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BASE_ADDR												0	0	0	0
W																
Reset	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-1. Internal Memory Map Registers' Base Address Register (IMMRBAR)

Table 2-3. IMMRBAR field descriptions

Field	Description
BASE_ADDR	Identifies the 12 most-significant address bits of the base of the 1 MB internal memory window.

Table 2-4. MPC5125 memory map

Offset from IMMRBAR (0xFF40_0000)	Absolute offset[1]	Region Name	Section/Page
0x0_0000–0_01FF	0xFF40_0000	System configuration (XLBMEN)	Chapter 2/2-7
0x0_0200–0_08FF	0xFF40_0200	Reserved[2]	
0x0_0900–0_09FF	0xFF40_0900	Software watchdog timer (WDT)	Chapter 29/29-799
0x0_0A00–0_0AFF	0xFF40_0A00	Real time clock (RTC)	Chapter 27/27-423
0x0_0B00–0_0BFF	0xFF40_0B00	General purpose timer 1 (GPT1)	Chapter 15/15-79
0x0_0C00–0_0CFF	0xFF40_0C00	Integrated programmable interrupt controller (IPIC)	Chapter 17/17-413
0x0_0D00–0_0DFF	0xFF40_0D00	CSB arbiter	Chapter 8/8-67
0x0_0E00–0_0EFF	0xFF40_0E00	Reset module (RESET)	Chapter 4/4-61

Table 2-4. MPC5125 memory map (continued)

Offset from IMMRBAR (0xFF40_0000)	Absolute offset[1]	Region Name	Section/Page
0x0_0F00–0_0FFF	0xFF40_0F00	Clock module (CLOCK)	Chapter 5/5-75
0x0_1000–0_10FF	0xFF40_1000	Power management control (PMC)	Chapter 24/24-337
0x0_1100–0_117F	0xFF40_1100	General Purpose I/O (GPIO1)	Chapter 16/16-97
0x0_1180–0_11FF	0xFF40_1180	General Purpose I/O (GPIO2)	Chapter 16/16-97
0x0_1200–0_12FF	0xFF40_1200	Reserved	
0x0_1300–0_137F	0xFF40_1300	MSCAN1	Chapter 22/22-249
0x0_1380–0_13FF	0xFF40_1380	MSCAN2	Chapter 22/22-249
0x0_1400–0_14FF	0xFF40_1400	Byte data link controller (BDLC)	Chapter 6/6-111
0x0_1500–0_15FF	0xFF40_1500	Secure digital host controller (SDHC1)	Chapter 28/28-447
0x0_1600–0_16FF	0xFF40_1600	Reserved	
0x0_1700–0_171F	0xFF40_1700	Inter-integrated circuit (I^2C1)	Chapter 19/19-161
0x0_1720–0_173F	0xFF40_1720	Inter-integrated circuit (I^2C2)	Chapter 19/19-161
0x0_1740–0_17FF	0xFF40_1740	Inter-integrated circuit (I^2C3)	Chapter 19/19-161
0x0_1800–0_1FFF	0xFF40_1800	Reserved	
0x0_2000–0_20FF	0xFF40_2000	Reserved	
0x0_2100–0_21FF	0xFF40_2100	Display Interface Unit (DIU)	Chapter 10/10-133
0x0_2200–0_22FF	0xFF40_2200	Reserved	
0x0_2300–0_237F	0xFF40_2300	MSCAN3	Chapter 22/22-249
0x0_2380–0_23FF	0xFF40_2380	MSCAN4	Chapter 22/22-249
0x0_2400–0_27FF	0xFF40_2400	Reserved	
0x0_2800–0_2FFF	0xFF40_2800	Fast Ethernet Controller (FEC1)	Chapter 14/14-25
0x0_3000–0_33FF	0xFF40_3000	USB ULPI1	Chapter 33/33-57
0x0_3400–0_3FFF	0xFF40_3400	Reserved	

Table 2-4. MPC5125 memory map (continued)

Offset from IMMRBAR (0xFF40_0000)	Absolute offset[1]	Region Name	Section/Page
0x0_4000–0_43FF	0xFF40_4000	USB ULPI2	Chapter 33/33-57
0x0_4400–0_47FF	0xFF40_4400	Reserved	
0x0_4800–0_4FFF	0xFF40_4800	Fast Ethernet Controller (FEC2)	Chapter 14/14-25
0x0_5000–0_50FF	0xFF40_5000	General purpose timer 2 (GPT2)	Chapter 15/15-79
0x0_5100–0_51FF	0xFF40_5100	Secure digital host controller (SDHC2)	Chapter 28/28-447
0x0_5200–0_8FFF	0xFF40_5200	Reserved	
0x0_9000–0_9FFF	0xFF40_9000	Multi-port DRAM controller (MDDRC)	Chapter 11/11-179
0x0_A000–0_AFFF	0xFF40_A000	I/O control	Chapter 20/20-495
0x0_B000–0_BFFF	0xFF40_B000	IC identification module (IIM)	Chapter 18/18-149
0x0_C000–0_FFFF	0xFF40_C000	Reserved	
0x1_0000–1_01FF	0xFF41_0100	LocalPlus controller (LPC)	Chapter 21/21-211
0x1_0200–1_0FFF	0xFF41_0200	Reserved	
0x1_1000–1_10FF	0xFF41_1000	Programmable Serial Controller 0 (PSC0)	Chapter 25/25-659
0x1_1100–1_11FF	0xFF41_1100	Programmable Serial Controller 1 (PSC1)	Chapter 25/25-659
0x1_1200–1_12FF	0xFF41_1200	Programmable Serial Controller 2 (PSC2)	Chapter 25/25-659
0x1_1300–1_13FF	0xFF41_1300	Programmable Serial Controller 3 (PSC3)	Chapter 25/25-659
0x1_1400–1_14FF	0xFF41_1400	Programmable Serial Controller 4 (PSC4)	Chapter 25/25-659
0x1_1500–1_15FF	0xFF41_1500	Programmable Serial Controller 5 (PSC5)	Chapter 25/25-659
0x1_1600–1_16FF	0xFF41_1600	Programmable Serial Controller 6 (PSC6)	Chapter 25/25-659
0x1_1700–1_17FF	0xFF41_1700	Programmable Serial Controller 7 (PSC7)	Chapter 25/25-659
0x1_1800–1_18FF	0xFF41_1800	Programmable Serial Controller 8 (PSC8)	Chapter 25/25-659

Table 2-4. MPC5125 memory map (continued)

Offset from IMMRBAR (0xFF40_0000)	Absolute offset[1]	Region Name	Section/Page
0x1_1900–1_19FF	0xFF41_1900	Programmable Serial Controller 9 (PSC9)	Chapter 25/25-659
0x1_1A00–1_1EFF	0xFF41_1A00	Reserved	
0x1_1F00–1_1FFF	0xFF41_1F00	Serial FIFO (SFIFO) for PSC 0 to 9	Chapter 25/25-659
0x1_2000–1_3FFF	0xFF41_2000	Reserved	
0x1_4000–1_57FF	0xFF41_4000	DMA	Chapter 9/9-87
0x1_5800–F_FFFF	0xFF41_5800	Reserved	

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000.

2 Reserved locations are not guaranteed to be empty. Software should not access reserved locations.

2.2.5.1.3 LocalPlus Boot Access Window Register (LPBAW)

The LocalPlus Access Window register (LPBAW) is shown in Figure 2-2.

Address: Base + 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	START_ADDR															
Reset[1]	See Table 2-6															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	STOP_ADDR															
Reset	See Table 2-6															

Figure 2-2. LocalPlus Boot Access Window Registers (LPBAW)

1 1. The LPBAW reset value depends on the Reset Configuration Word High (RCWH) values. See Table 2-6.

Table 2-5. LPBAW field descriptions

Field	Description
START_ADDR	Any access on an address between Start and Stop Address enables the corresponding chip select. The START_ADDR is for the address comparison extended with 0x0000. The STOP_ADDR is for the address comparison extended with 0xFFFF. This means the minimum address size is 64 KB. If the START_ADDR and STOP_ADDR are set to 0xA000, the access window goes from 0xA000_0000 to 0xA000_FFFF. **Note:** CS Boot and CS0 have the same physical CS pin.
STOP_ADDR	

The Power Architecture core may fetch its boot vector from a local bus peripheral device. For this purpose, LPBAW[START_ADDR] and LPBAW[STOP_ADDR] reset values are set according to the value of the BMS and ROM_LOC bits in the Reset Configuration Word High (RCWH) register.

Table 2-6 defines the reset values for the START_ADDR and STOP_ADDR bitfields in the LPBAW and LPCS*n*AW registers.

Table 2-6. LPBAW/LPCS*n*AW [START_ADDR] and [STOP_ADDR] Reset Values

ROM_LOC	BMS	START_ADDR Reset Value	STOP_ADDR Reset Value
00	0	0x0000	0x007F
00	1	0xFF80	0xFFFF
01	*x*	0x0100	0x0100

NOTE

BMS is bit 26 of the reset configuration word high register (RCWHR). Its initial value is set by the state of the EMB_AD02 pin at the release of $\overline{\text{PORESET}}$. ROM_LOC is set by bits 22 and 21 of the reset configuration word high register (RCWHR). The initial values of these bits are set by the states of the EMB_AD0 and EMB_AD1 pins at the release of $\overline{\text{PORESET}}$. See Figure 2-8.

2.2.5.1.4 LocalPlus CS0–7 Access Window Registers (LPCS*x*AW)

The LocalPlus Access Window Registers (LPCS*x*AW) are shown in Figure 2-2.

Address: Base + 0x0024 (LPCS0AW) Base + 0x0034 (LPCS4AW)
Base + 0x0028 (LPCS1AW) Base + 0x0038 (LPCS5AW)
Base + 0x002C (LPCS2AW) Base + 0x003C (LPCS6AW)
Base + 0x0030 (LPCS3AW) Base + 0x0040 (LPCS7AW)
Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	START_ADDR															
Reset	See Table 2-6															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	STOP_ADDR															
Reset	See Table 2-6															

Figure 2-3. LocalPlus CS0–7 Access Window Registers (LPCSxAW)

Table 2-7. LPCSxAW field descriptions

Field	Description
START_ADDR	Any access on an address between Start and Stop Address enables the corresponding chip select. The START_ADDR is for the address comparison extended with 0x0000. The STOP_ADDR is for the address comparison extended with 0xFFFF. This means the minimum address size is 64 k. If the START_ADDR and STOP_ADDR are set to 0xA000, the access window goes from 0xA000_0000 to 0xA000_FFFF. **Note:** CS Boot and CS0 have the same physical CS pin.
STOP_ADDR	

2.2.5.1.5 DDR Local Access Window Base Address Register (DDRLAWBAR)

Figure 2-4 shows the DDR Local Access Window Base Address Register (DDRLAWBAR).

Address: Base + 0x00A0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BASE_ADDR															
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BASE_ADDR				0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-4. DDR Local Access Window Base Address Register (DDRLAWBAR)

Table 2-8. DDRLAWBAR0 field descriptions

Field	Description
BASE_ADDR	Identifies the 20 most-significant address bits of the base address of local access window n. The specified base address should be aligned to the window size, as defined by DDRLAWAR[SIZE].

2.2.5.1.6 DDR Local Access Window Attributes Register (DDRLAWAR)

Figure 2-5 shows the DDR Local Access Window Attributes Register (DDRLAWAR).

Address: Base + 0x00A4 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	SIZE					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1

Figure 2-5. DDR Local Access Window Attributes Register (DDRLAWAR)

Table 2-9. DDRLAWAR field descriptions

Field	Description
SIZE	Identifies the size of the window from the starting address. Window size is $2^{(SIZE+1)}$ bytes. 000000–011000 Reserved. Window is undefined. 011001 64 MB 011010 128 MB 011011 256 MB .... $2^{(SIZE + 1)}$ bytes ..... 011110 2 GB 011111–111111 Reserved. Window is undefined.

2.2.5.1.7 SRAM Base Address Register (SRAMBAR)

Figure 2-6 shows the SRAM Base Address Register (SRAMBAR).

Address: Base + 0x00C4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BASE_ADDR														0	0
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-6. SRAM Base Address Register (SRAMBAR)

Table 2-10. SRAMBAR field descriptions

Field	Description
BASE_ADDR	Identifies the 14 most-significant address bits of the base address of the 256 KB SRAM memory window.

NOTE

Although 256 KB memory space is reserved, the real SRAM size is 32 KB. Any address in a 256 KB window is directed into 32 KB memory by modulo 32 KB.

2.2.5.1.8 NFC Base Address Register (NFCBAR)

Figure 2-7 shows the NFC Base Address Register (NFCBAR). The NFCBAR[BASE_ADDR] reset value depends on the reset configuration word. See Section 2.2.5.1.9, “NFCBAR[BASE_ADDR] Reset Value,” for a detailed description.

Address: Base + 0x00C8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BASE_ADDR												0	0	0	0
W																
Reset	See Table 2-12												0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-7. NFC Base Address Registers (NFCBAR)

Table 2-11. NFCBAR field descriptions

Field	Description
BASE_ADDR	Identifies the 12 most-significant address bits of the base address of the 1 MB NFC memory window.

2.2.5.1.9 NFCBAR[BASE_ADDR] Reset Value

The Power Architecture core may use a local bus peripheral device to fetch its boot vector. For this purpose, NFCBAR[BASE_ADDR] reset value is set according to the value set in the reset configuration word high BMS and ROM_LOC fields.

Table 2-12 defines the reset value NFCBAR[BASE_ADDR].

Table 2-12. NFCBAR[BASE_ADDR] Reset Value

ROM_LOC	BMS	BASE_ADDR Reset Value
01	0	0x000
01	1	0xFFF
00	*x*	0x400

NOTE

BMS is bit 26 of the reset configuration word high register (RCWHR). Its initial value is set by the state of the EMB_AD02 pin at the release of PORESET. ROM_LOC is set by bits 22 and 21 of the reset configuration word high register (RCWHR). The initial values of these bits are set by the states of the EMB_AD00 and EMB_AD01 pins at the release of PORESET.

See Table 2-13 for initial memory map values based on the EMBAD0/1 and BMS bits.

Table 2-13. Memory Map Values based on EMBAD0/1 and BMS

ROM_LOC	BMS	BOOT START	BOOT_STOP	NFC_BASE_ADDR	NFC_STOP_ADDR
00	0	0x0000 0000	0x007F FFFF	0x4000 0000	0x400F FFFF
00	1	0xFF80 0000	0xFFFF FFFF	0x4000 0000	0x400F FFFF
01	0	0x0100 0000	0x0100 FFFF	0x0000 0000	0x000F FFFF
01	1	0x0100 0000	0x0100 FFFF	0xFFF0 0000	0xFFFF FFFF

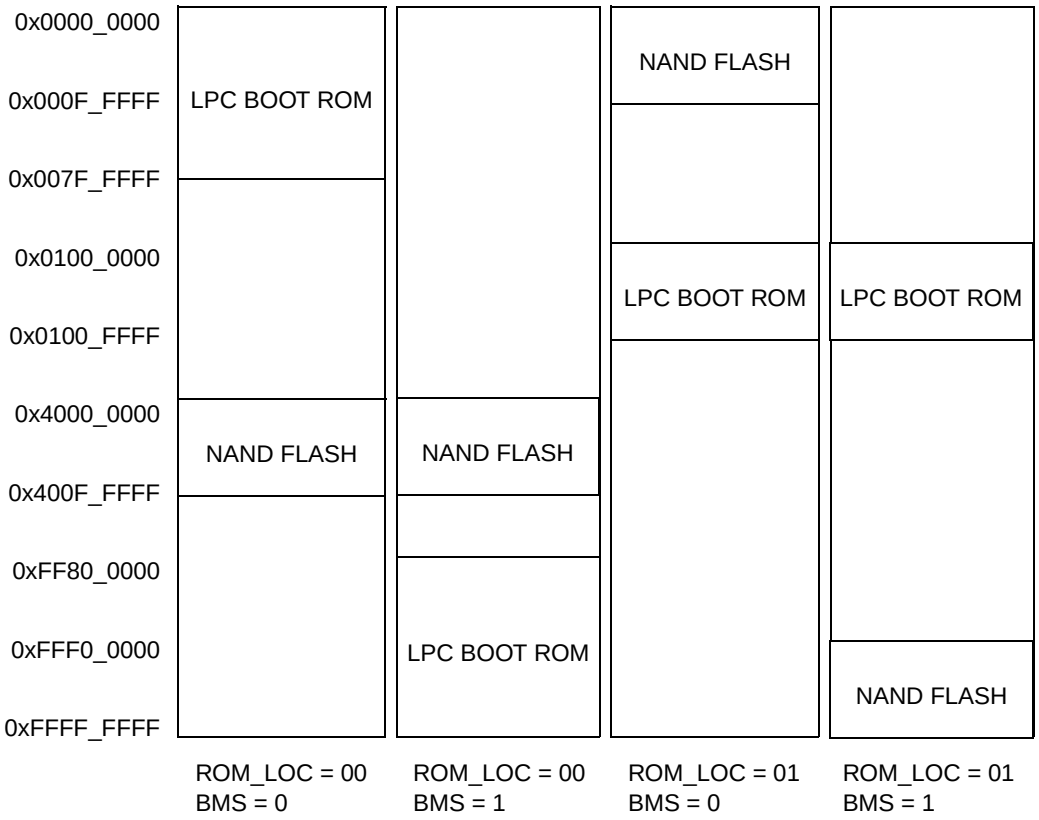


Figure 2-8. Initial Memory Map Configurations Immediately After the Release of $\overline{\text{PORESET}}$

2.2.6 Overlap of Local Access Windows

It is generally a programming error to overlap the addressing range of the Local Access Windows. Nothing prevents software from programming overlapping addresses.

2.2.7 Configuring Local Access Windows

After a local access window is enabled, it should not be modified while any device in the system may be using the window. Neither should a new window be used until the effect of the write to the window is visible to all blocks that use the window. This can be guaranteed by completing a read of the last local

access window configuration register before enabling any other devices to use the window. For instance, if LPC local access windows 1–3 are being configured in order during the initialization process, the last write (to LPCS2AW) should be followed by a read of LPCS2AW before any devices try to use any of these windows. If the configuration is being done by the local Power Architecture processor, the read of LPCS2AW should be followed by an **isync** instruction.

2.3 System Configuration

Some general information and configuration options that affect the system behavior and performance are described in the following sections.

2.3.1 System Configuration Register Memory Map

Table 2-14 shows the memory map for the system configuration registers.

Table 2-14. System Configuration Register Memory Map

Local Memory Offset (Hex)	Register	Access	Reset value	Section/Page
0x100	System Part and Revision ID Register (SPRIDR)	R	0x8019_0010	2.3.1.1.1/2-21
0x104	System Priority Configuration Register (SPCR)	R/W	0x0000_0010	2.3.1.1.2/2-22
0x108–0x1FC	Reserved			

2.3.1.1 System Configuration Registers

2.3.1.1.1 System Part and Revision ID Register (SPRIDR)

The System Part and Revision ID Register shown in Figure 2-9 provides information about the part and revision numbers.

Address: Base + 0x0100 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PARTID															
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REVID															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 2-9. System Part and Revision ID Register (SPRIDR)

Table 2-15. SPRIDR field descriptions

Field	Description
PARTID	Part Identification. This read-only field is mask-programmed with a code corresponding to the part number. It is intended to help factory test and user code which is sensitive to part changes. The part number changes according to manufacturing considerations.
REVID	Revision Identification. This read-only field is mask-programmed with a code corresponding to the revision number of the part defined in PARTID field. It is intended to help factory test and user code which is sensitive to part changes. The mask number changes with each mask set change.

NOTE

PARTID and REVID are specific for the e300 SVR value.

2.3.1.1.2 System Priority Configuration Register (SPCR)

The System Priority Configuration Register shown in Figure 2-10 controls the priority of requests for transactions on the internal system bus. This priority is considered by the system arbiter when an internal unit requests the bus.

Address: Base + 0x0104 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	TBEN	COREPR		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	SAPPR		0	0	0	0	0	0	0	NFCBOOTEN	TEMPPD	TEMPSEL		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 2-10. System Priority Configuration Register (SPCR)

Table 2-16. SPCR field descriptions

Field	Description
TBEN	Power Architecture Core time base unit enable 0 Time base unit is disabled. 1 Time base unit is enabled.
COREPR	Power Architecture Core system bus request priority. The level of priority can be chosen from four possible levels. 00 Level 0 (Lowest Priority). 01 Level 1. 10 Level 2. 11 Level 3 (Highest Priority)
SAPPR	SAP system bus request priority. The level of priority can be chosen from four possible levels. 00 Level 0 (Lowest Priority). 01 Level 1. 10 Level 2. 11 Level 3 (Highest Priority).

Table 2-16. SPCR field descriptions (continued)

Field	Description
NFCBOOTEN	NFC Boot after Soft Reset 0 No NFC boot after soft reset even if ROM_LOC = 01. 1 NFC boot after soft reset if ROM_LOC = 01.
TEMPPD	Temperature Sensor 0 Temperature Sensor is enabled. 1 Temperature Sensor is disabled.
TEMPSEL	Temperature Sensor select trip point 000 105 °C 001 95 °C 010 85 °C 011 75 °C 100 65 °C 101 55 °C 110 45 °C 111 35 °C

This page is intentionally left blank.

Chapter 3
Signal Descriptions

3.1 Introduction

This chapter describes the MPC5125 external signals. It is organized into the following sections:

- Overview of signals and cross references for signals that serve multiple functions, including two lists: one ordered by functional block and one by alphabetical
- List of reset configuration signals
- List of output signal states during reset

NOTE

A bar over a signal name indicates that the signal is active low, such as $\overline{\text{HRESET}}$ (Hard Reset). Active-low signals are referred to as asserted (active) when they are low and negated when they are high. Signals that are not active low, such as MODT (DDR2 on-die-termination output), are referred to as asserted when they are high and negated when they are low.

3.1.1 Signals Overview

The MPC5125 signals are grouped as follows:

- DDR memory interface signals
- PSC interface signals
- I^2C interface signals
- LPC interface signals
- NFC interface signals
- EMB interface signals
- DIU interface signals
- USB ULPI interface signals
- ETH interface signals
- CAN interface signals
- J1850 interface signals
- JTAG, test, system control signals
- Clock signals

Refer to the *MPC5125 Microcontroller Data Sheet* at www.freescale.com for a pinout diagram showing pin numbers and a listing of all the electrical and mechanical specifications.

Functionality, which is not a primary function, is not shown in Figure 3-1. These functions are multiplexed. Multiplexing is shown in Table 3-1.

Individual chapters of this document provide details for each signal, describing each signal's behavior when asserted and negated and when the signal is an input or an output. Power signals are described in the *MPC5125 Microcontroller Data Sheet*.

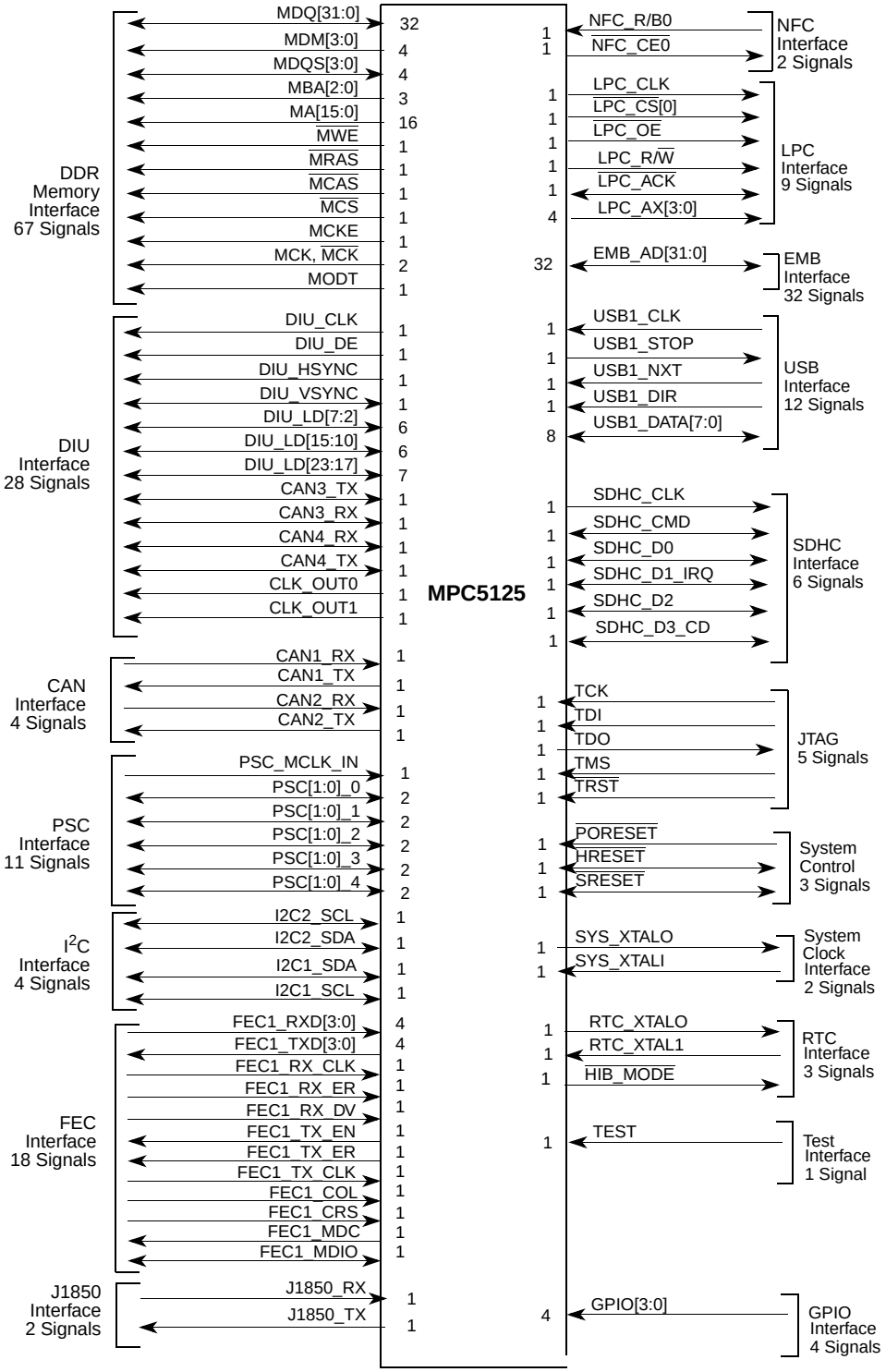


Figure 3-1. MPC5125 Pin Signal Groupings

Table 3-1 provides a summary of pin multiplexing for the MPC5125.

Table 3-1. MPC5125 Pin Multiplexing

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
GPIO00	—	ALT0 ALT1 ALT2 ALT3	GPIO00 — — —	GPIO1 — — —	I — — —	VBAT	Dedicated input can be used to receive an external wakeup.	B6
GPIO01	—	ALT0 ALT1 ALT2 ALT3	GPIO01 — — —	GPIO1 — — —	I — — —	VBAT	Dedicated input can be used to receive an external wakeup.	A5
GPIO02	—	ALT0 ALT1 ALT2 ALT3	GPIO02 — — —	GPIO1 — — —	I — — —	VBAT	Dedicated input can be used to receive an external wakeup.	A6
GPIO03	—	ALT0 ALT1 ALT2 ALT3	GPIO03 — — —	GPIO1 — — —	I — — —	VBAT	Dedicated input can be used to receive an external wakeup.	C7
RTC_XTALI	—	ALT0 ALT1 ALT2 ALT3	RTC_XTALI — — —	RTC — — —	I — — —	VBAT	—	A8
RTC_XTALO	—	ALT0 ALT1 ALT2 ALT3	RTC_XTALO — — —	RTC — — —	O — — —	VBAT	—	A7
$\overline{\text{HIB\_MODE}}$	—	ALT0 ALT1 ALT2 ALT3	$\overline{\text{HIB\_MODE}}$ — — —	RTC — — —	O — — —	VBAT	In Hibernation mode , this pin provides a signal to shut down an external power supply.	C8
Analog Visible Signal								
SPLL_ANAVIZ	—	ALT0 ALT1 ALT2 ALT3	SPLL_ANAVIZ — — —	 — — —	 — — —	—	—	D9

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
TMPS_ANAVIZ	—	ALT0 ALT1 ALT2 ALT3	TMPS_ANAVIZ — — —	 — — —	 — — —	—	—	D1
SYS_XTALI	—	ALT0 ALT1 ALT2 ALT3	SYS_XTALI — — —	SysClock — — —	I — — —	SYS_PLL _AVDD	—	A9
SYS_XTALO	—	ALT0 ALT1 ALT2 ALT3	SYS_XTALO — — —	SysClock — — —	O — — —	SYS_PLL _AVDD	—	A10
$\overline{\text{MCS}}$	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	$\overline{\text{MCS0}}$ — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	D18
$\overline{\text{MCAS}}$	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	$\overline{\text{MCAS}}$ — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	A20
$\overline{\text{MRAS}}$	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	$\overline{\text{MRAS}}$ — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	C19
MVREF	—	ALT0 ALT1 ALT2 ALT3	MVREF — — —	DRAM — — —	I — — —	VDD_IO_MEM	—	N19
MVTT0	—	ALT0 ALT1 ALT2 ALT3	MVTT0 — — —	DRAM — — —	I — — —	VDD_IO_MEM	—	W18

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
MVTT1	—	ALT0 ALT1 ALT2 ALT3	MVTT1 — — —	DRAM — — —	I — — —	VDD_IO_MEM	—	R19
MVTT2	—	ALT0 ALT1 ALT2 ALT3	MVTT2 — — —	DRAM — — —	I — — —	VDD_IO_MEM	—	M19
MVTT3	—	ALT0 ALT1 ALT2 ALT3	MVTT3 — — —	DRAM — — —	I — — —	VDD_IO_MEM	—	K19
$\overline{\text{MWE}}$	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	$\overline{\text{MWE}}$ — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	A21
MDQ00	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ00 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	AB19
MDQ01	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ01 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	Y18
MDQ02	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ02 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	AA19
MDQ03	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ03 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	AB21

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
MDQ04	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ04 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	AA21
MDQ05	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ05 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	Y20
MDQ06	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ06 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	W20
MDQ07	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ07 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	U19
MDQ08	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ08 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	AA22
MDQ09	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ09 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	Y22
MDQ10	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ10 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	V20
MDQ11	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ11 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	W21

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
MDQ12	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ12 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	W22
MDQ13	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ13 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	T20
MDQ14	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ14 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	V22
MDQ15	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ15 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	U22
MDQ16	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ16 — — GPT1[0]	DRAM — — GPT1	I/O — — I/O	VDD_IO_MEM	—	R20
MDQ17	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ17 — — GPT1[1]	DRAM — — GPT1	I/O — — I/O	VDD_IO_MEM	—	T21
MDQ18	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ18 — — GPT1[2]	DRAM — — GPT1	I/O — — I/O	VDD_IO_MEM	—	P20
MDQ19	0x00 IO_CON- TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ19 — — GPT1[3]	DRAM — — GPT1	I/O — — I/O	VDD_IO_MEM	—	T22

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
MDQ20	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ20 — — GPT1[4]	DRAM — — GPT1	I/O — — I/O	VDD_IO_MEM	—	N20
MDQ21	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ21 — — GPT1[5]	DRAM — — GPT1	I/O — — I/O	VDD_IO_MEM	—	P22
MDQ22	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ22 — — GPT1[6]	DRAM — — GPT1	I/O — — I/O	VDD_IO_MEM	—	N22
MDQ23	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ23 — — GPT1[7]	DRAM — — GPT1	I/O — — I/O	VDD_IO_MEM	—	M20
MDQ24	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ24 — — GPIO21	DRAM — — GPIO1	I/O — — I/O	VDD_IO_MEM	—	M21
MDQ25	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ25 — — GPIO22	DRAM — — GPIO1	I/O — — I/O	VDD_IO_MEM	—	M22
MDQ26	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ26 — — GPIO23	DRAM — — GPIO1	I/O — — I/O	VDD_IO_MEM	—	L20
MDQ27	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ27 — — GPIO24	DRAM — — GPIO1	I/O — — I/O	VDD_IO_MEM	—	L21

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
MDQ28	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ28 — — GPIO25	DRAM — — GPIO1	I/O — — I/O	VDD_IO_MEM	—	K20
MDQ29	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ29 — — GPIO26	DRAM — — GPIO1	I/O — — I/O	VDD_IO_MEM	—	J22
MDQ30	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ30 — — GPIO27	DRAM — — GPIO1	I/O — — I/O	VDD_IO_MEM	—	J21
MDQ31	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQ31 — — GPIO28	DRAM — — GPIO1	I/O — — I/O	VDD_IO_MEM	—	J20
MDM0	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDM0 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	Y19
MDM1	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDM1 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	V21
MDM2	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDM2 — — GPIO29	DRAM — — GPIO1	O — — I/O	VDD_IO_MEM	—	R22
MDM3	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDM3 — — GPIO30	DRAM — — GPIO1	O — — I/O	VDD_IO_MEM	—	K22

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
MDQS0	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQS0 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	AA20
MDQS1	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQS1 — — —	DRAM — — —	I/O — — —	VDD_IO_MEM	—	U20
MDQS2	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQS2 — — GPIO31	DRAM — — GPIO1	I/O — — I/O	VDD_IO_MEM	—	P21
MDQS3	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MDQS3 — — GPIO32	DRAM — — GPIO2	I/O — — I/O	VDD_IO_MEM	—	L22
MBA0	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MBA0 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	H19
MBA1	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MBA1 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	H20
MBA2	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MBA2 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	G21
MA00	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA00 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	G20

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
MA01	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA01 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	F22
MA02	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA02 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	F19
MA03	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA03 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	E22
MA04	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA04 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	E21
MA05	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA05 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	F20
MA06	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA06 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	D22
MA07	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA07 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	E20
MA08	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA08 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	D21

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
MA09	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA09 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	C22
MA10	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA10 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	E19
MA11	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA11 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	B22
MA12	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA12 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	C20
MA13	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA13 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	D20
MA14	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA14 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	B21
MA15	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MA15/MCS1 — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	B20
MCK	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MCK — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	H22

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
$\overline{\text{MCK}}$	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	$\overline{\text{MCK}}$ — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	G22
MCKE	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MCKE — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	C18
MODT	0x00 IO_CON-TROL_MEM	ALT0 ALT1 ALT2 ALT3	MODT — — —	DRAM — — —	O — — —	VDD_IO_MEM	—	J19
LPC_CLK	0x04 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_CLK TPA1 — GPIO04	LPC — — GPIO1	O — I/O	VDD_IO	—	R4
LPC_OE_B	0x05 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_OE PSC3_3 — GPIO05	LPC PSC3 — GPIO1	O I/O — I/O	VDD_IO	—	N2
LPC_RWB	0x06 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_R/$\overline{\text{W}}$ PSC3_4 — GPIO06	LPC PSC3 — GPIO1	O I/O — I/O	VDD_IO	—	N3
LPC_CS0_B	0x07 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_CS0 — — GPIO07	LPC — — GPIO1	O — — I/O	VDD_IO	—	M1
LPC_ACK_B	0x08 STD_PU	ALT0 ALT1 ALT2 ALT3	$\overline{\text{LPC\_ACK}}$/$\overline{\text{LPC\_BURST}}$ NFC_CE1 LPC_CS1 GPIO08	LPC NFC LPC GPIO1	I/O O O I/O	VDD_IO	—	P3

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
LPC_AX03	0x09 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AX03/LPC_TS NFC_CE2 LPC_CS2 —	LPC NFC LPC —	O O O —	VDD_IO	—	L1
EMB_AD00	0x2C STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD00/NFC_AD00 — RST_CONF_LOC0 —	LPC — —	I/O — —	VDD_IO	ALT2: Reset configuration Boot ROM Location 0	A4
EMB_AD01	0x2B STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD01/NFC_AD01 — RST_CONF_LOC1 —	LPC — —	I/O — —	VDD_IO	ALT2: Reset configuration Boot ROM Location 1	A3
EMB_AD02	0x2A STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD02/NFC_AD02 — RST_CONF_BMS —	LPC — —	I/O — —	VDD_IO	ALT2: Reset configuration Boot Mode Select	B4
EMB_AD03	0x29 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD03/NFC_AD03 — RST_CONF_LPCDBW0 —	LPC — —	I/O — —	VDD_IO	ALT2: Reset configuration LPC Port Size 0	B3
EMB_AD04	0x28 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD04/NFC_AD04 — RST_CONF_LPCDBW1 —	LPC — —	I/O — —	VDD_IO	ALT2: Reset configuration LPC Port Size 1	D5
EMB_AD05	0x27 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD05/NFC_AD05 — RST_CONF_COREPLL6 —	LPC — —	I/O — —	VDD_IO	ALT2: Reset configuration Core PLL Multiplication Factor 0	B2
EMB_AD06	0x26 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD06/NFC_AD06 — RST_CONF_COREPLL5 —	LPC — —	I/O — —	VDD_IO	ALT2: Reset configuration Core PLL Multiplication Factor 1	C4

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
EMB_AD07	0x25 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD07/NFC_AD07 — RST_CONF_COREPLL4 —	LPC — —	I/O — —	VDD_IO	ALT2: Reset configuration Core PLL Multiplication Factor 2	C3
EMB_AD08	0x24 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD08/NFC_AD08 PSC3_2 RST_CONF_SPMF0 GPIO28	LPC PSC3 GPIO1	I/O I/O I/O	VDD_IO	ALT2: Reset configuration System PLL Multiplication Factor 0	E4
EMB_AD09	0x23 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD09/NFC_AD09 PSC3_1 RST_CONF_SPMF1 GPIO27	LPC PSC3 GPIO1	I/O I/O I/O	VDD_IO	ALT2: Reset configuration System PLL Multiplication Factor 1	C2
EMB_AD10	0x22 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD10/NFC_AD10 PSC3_0 RST_CONF_SPMF2 GPIO26	LPC PSC3 GPIO1	I/O I/O I/O	VDD_IO	ALT2: Reset configuration System PLL Multiplication Factor 2	D2
EMB_AD11	0x21 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD11/NFC_AD11 PSC2_4 RST_CONF_SPMF3 GPIO25	LPC PSC2 GPIO1	I/O I/O I/O	VDD_IO	ALT2: Reset configuration	C1
EMB_AD12	0x20 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD12/NFC_AD12 PSC2_3 RST_CONF_PREDIV0 GPIO24	LPC PSC2 GPIO1	I/O I/O I/O	VDD_IO	ALT2: Reset configuration	E3
EMB_AD13	0x1F STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD13/NFC_AD13 PSC2_2 RST_CONF_PREDIV1 GPIO23	LPC PSC2 GPIO1	I/O I/O I/O	VDD_IO	ALT2: Reset configuration	E2
EMB_AD14	0x1E STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD14/NFC_AD14 PSC2_1 RST_CONF_PREDIV2 GPIO22	LPC PSC2 GPIO1	I/O I/O I/O	VDD_IO	ALT2: Reset configuration	H4

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
EMB_AD15	0x1D STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD15/NFC_AD15 PSC2_0 RST_CONF_SYSOSCEN GPIO21	LPC PSC2 GPIO1	I/O I/O I/O	VDD_IO	ALT2: Reset configuration	E1
EMB_AD16	0x1C STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD16/LPC_A01/$\overline{\text{NFC\_WE}}$ — — —	LPC — —	I/O — —	VDD_IO	—	F3
EMB_AD17	0x1B STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD17/LPC_A02/$\overline{\text{NFC\_RE}}$ — RST_CONF_PLL_LOCK —	LPC — —	I/O — —	VDD_IO	ALT2: Reset configuration	G3
EMB_AD18	0x1A STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD18/LPC_A03/NFC_CLE — RST_CONF_LPCMX —	LPC — —	I/O — —	VDD_IO	ALT2: Reset configuration	G2
EMB_AD19	0x19 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD19/LPC_A04/NFC_ALE — RST_CONF_LPCWA —	LPC — —	I/O — —	VDD_IO	ALT2: Reset configuration	J4
EMB_AD20	0x18 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD20/LPC_A05 — — GPIO20	LPC GPIO1	I/O — — I/O	VDD_IO	—	H3
EMB_AD21	0x17 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD21/LPC_A06 — — GPIO19	LPC GPIO1	I/O — — I/O	VDD_IO	—	F1

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
EMB_AD22	0x16 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD22/LPC_A07 — RST_CONF_LPC_TS GPIO18	LPC GPIO1	I/O — I/O	VDD_IO	ALT2: Reset configuration	K4
EMB_AD23	0x15 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD23/LPC_A08 — — GPIO17	LPC GPIO1	I/O — — I/O	VDD_IO	—	J3
EMB_AD24	0x14 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD24/LPC_A09 — — GPIO16	LPC GPIO1	I/O — — I/O	VDD_IO	—	K3
EMB_AD25	0x13 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD25/LPC_A10 — — GPIO15	LPC GPIO1	I/O — — I/O	VDD_IO	—	G1
EMB_AD26	0x12 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD26/LPC_A11 — — GPIO14	LPC GPIO1	I/O — — I/O	VDD_IO	—	J2
EMB_AD27	0x11 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD27/LPC_A12 — — GPIO13	LPC GPIO1	I/O — — I/O	VDD_IO	—	M4
EMB_AD28	0x10 STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD28/LPC_A13 — — GPIO12	LPC GPIO1	I/O — — I/O	VDD_IO	—	H1
EMB_AD29	0x0F STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AD29/LPC_A14 — — GPIO11	LPC GPIO1	I/O — — I/O	VDD_IO	—	L3

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
EMB_AD30	0x0E STD_PU_ST	ALT0 ALT1 ALT2 ALT3	LPC_AD30/LPC_A15 CAN_CLK — GPIO10	LPC GPIO1	I/O O — I/O	VDD_IO	—	M3
EMB_AD31	0x0D STD_PU_ST	ALT0 ALT1 ALT2 ALT3	LPC_AD31/LPC_A16 PSC_MCLK_IN — GPIO09	LPC GPIO1	I/O I — I/O	VDD_IO	—	J1
EMB_AX00	0x0C STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AX00/LPC_ALE — — —	LPC — — —	O — — —	VDD_IO	—	K1
EMB_AX01	0x0B STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AX01/LPC_TSIZ0 — LPC_CS4 —	LPC — LPC —	O — O —	VDD_IO	—	N4
EMB_AX02	0x0A STD_PU	ALT0 ALT1 ALT2 ALT3	LPC_AX02/LPC_TSIZ1 NFC_CE3 LPC_CS3 —	LPC LPC —	O O O —	VDD_IO	—	L2
NFC_CE0_B	0x02D STD_PU	ALT0 ALT1 ALT2 ALT3	NFC_CE0 — — GPIO29	NFC — — GPIO1	O — — I/O	VDD_IO	—	P1
NFC_RB	0x02E STD_PU_ST	ALT0 ALT1 ALT2 ALT3	NFC_R/B0 — — GPIO30	NFC — — GPIO1	I — — I/O	VDD_IO	When booting from the NFC, the NFC_RB pin needs an external pullup resistor.	N1
DIU_CLK	0x02F STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_CLK PSC4_0 USB1_DATA0 LPC_AX04	DIU PSC4 USB1 LPC	O I/O I/O O	VDD_IO	—	AB8

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
DIU_DE	0x030 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_DE PSC4_1 USB1_DATA1 LPC_AX05	DIU PSC4 USB1 LPC	O I/O I/O O	VDD_IO	—	AA8
DIU_HSYNC	0x031 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_HSYNC PSC4_2 USB1_DATA2 LPC_AX06	DIU PSC4 USB1 LPC	O I/O I/O O	VDD_IO	—	W11
DIU_VSYNC	0x032 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_VSYNC PSC4_3 USB1_DATA3 GPIO31	DIU PSC4 USB1 GPIO1	I/O I/O I/O I/O	VDD_IO	—	Y17
DIU_LD00	0x033 STD_PU_ST	ALT0 ALT1 ALT2 ALT3	CAN3_RX CLK_OUT2 DIU_LD00 GPIO32	CAN3 DIU DIU GPIO2	I O I/O I/O	VDD_IO	—	AB9
DIU_LD01	0x034 STD_PU	ALT0 ALT1 ALT2 ALT3	CAN3_TX CLK_OUT3 DIU_LD01 GPIO33	CAN3 DIU DIU GPIO2	O O I/O I/O	VDD_IO	—	Y10
DIU_LD02	0x035 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD02 PSC4_4 USB1_DATA4 LPC_AX07	DIU PSC4 USB1 LPC	 I/O I/O O	VDD_IO	—	AA10
DIU_LD03	0x036 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD03 PSC5_0 USB1_DATA5 LPC_AX08	DIU PSC5 USB1 LPC	I/O I/O I/O O	VDD_IO	—	Y11
DIU_LD04	0x037 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD04 PSC5_1 USB1_DATA6 LPC_AX09	DIU PSC5 USB1 LPC	I/O I/O I/O O	VDD_IO	—	AA11

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
DIU_LD05	0x038 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD05 PSC5_2 USB1_DATA7 GPIO34	DIU PSC5 USB1 GPIO2	I/O I/O I/O I/O	VDD_IO	—	AB10
DIU_LD06	0x039 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD06 PSC5_3 USB1_STOP GPIO35	DIU PSC5 USB1 GPIO2	I/O I/O O I/O	VDD_IO	—	AB11
DIU_LD07	0x03A STD_PU_ST	ALT0 ALT1 ALT2 ALT3	DIU_LD07 PSC5_4 USB1_CLK GPIO36	DIU PSC5 USB1 GPIO2	I/O I/O I I/O	VDD_IO	—	Y12
DIU_LD08	0x03B STD_PU_ST	ALT0 ALT1 ALT2 ALT3	CAN4_RX PSC6_0 DIU_LD08 GPIO37	CAN4 PSC6 DIU GPIO2	I I/O I/O I/O	VDD_IO	—	W13
DIU_LD09	0x03C STD_PU	ALT0 ALT1 ALT2 ALT3	CAN4_TX PSC6_1 DIU_LD09 GPIO38	CAN4 PSC6 DIU GPIO2	O I/O I/O I/O	VDD_IO	—	AB12
DIU_LD10	0x03D STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD10 PSC6_2 USB1_NEXT GPIO39	DIU PSC6 USB1 GPIO2	I/O I/O O I/O	VDD_IO	—	Y13
DIU_LD11	0x03E STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD11 PSC6_3 USB1_DIR GPIO40	DIU PSC6 USB1 GPIO2	I/O I/O I I/O	VDD_IO	—	AA13
DIU_LD12	0x03F STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD12 PSC6_4 USB2_DATA0 GPT2[0]	DIU PSC6 USB2 GPT2	I/O I/O I/O I/O	VDD_IO	—	AB13

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
DIU_LD13	0x040 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD13 PSC7_0 USB2_DATA1 GPT2[1]	DIU PSC7 USB2 GPT2	I/O I/O I/O I/O	VDD_IO	—	W14
DIU_LD14	0x041 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD14 PSC7_1 USB2_DATA2 GPT2[2]	DIU PSC7 USB2 GPT2	I/O I/O I/O I/O	VDD_IO	—	Y14
DIU_LD15	0x042 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD15 PSC7_2 USB2_DATA3 GPT2[3]	DIU PSC7 USB2 GPT2	I/O I/O I/O I/O	VDD_IO	—	AB14
DIU_LD16	0x043 STD_PU_ST	ALT0 ALT1 ALT2 ALT3	CLK_OUT0 I2C3_SCL DIU_LD16 GPIO41	DIU I^2C2 DIU GPIO2	O I/O I/O I/O	VDD_IO	—	AA15
DIU_LD17	0x044 STD_PU_ST	ALT0 ALT1 ALT2 ALT3	CLK_OUT1 I2C3_SDA DIU_LD17 GPIO42	DIU I^2C3 DIU GPIO2	O I/O I/O I/O	VDD_IO	—	Y15
DIU_LD18	0x045 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD18 PSC7_3 USB2_DATA4 GPT2[4]	DIU PSC7 USB2 GPT2	I/O I/O I/O I/O	VDD_IO	—	AB15
DIU_LD19	0x046 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD19 PSC7_4 USB2_DATA5 GPT2[5]	DIU PSC7 USB2 GPT2	I/O I/O I/O I/O	VDD_IO	—	AB16
DIU_LD20	0x047 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD20 PSC8_0 USB2_DATA6 GPT2[6]	DIU PSC8 USB2 GPT2	I/O I/O I/O I/O	VDD_IO	—	AB17

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
DIU_LD21	0x048 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD21 PSC8_1 USB2_DATA7 GPT2[7]	DIU PSC8 USB2 GPT2	I/O I/O I/O I/O	VDD_IO	—	W16
DIU_LD22	0x049 STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD22 PSC8_2 USB2_DIR GPIO43	DIU PSC8 USB2 GPIO2	I/O I/O I I/O	VDD_IO	—	Y16
DIU_LD23	0x04A STD_PU	ALT0 ALT1 ALT2 ALT3	DIU_LD23 PSC8_3 USB2_NEXT GPIO44	DIU PSC8 USB2 GPIO2	I/O I/O I I/O	VDD_IO	—	AA17
I2C2_SCL	0x4B STD_PU_ST	ALT0 ALT1 ALT2 ALT3	I2C2_SCL PSC8_4 USB2_CLK GPIO45	I^2C2 PSC8 USB2 GPIO2	I/O I/O I I/O	VDD_IO	—	V4
I2C2_SDA	0x4C STD_PU_ST	ALT0 ALT1 ALT2 ALT3	I2C2_SDA PSC9_4 USB2_STOP GPIO46	I^2C2 PSC9 USB2 GPIO2	I/O I/O O I/O	VDD_IO	—	U4
I2C1_SCL	0x4F STD_PU_ST	ALT0 ALT1 ALT2 ALT3	I2C1_SCL PSC9_2 CAN3_RX GPIO49	I^2C1 PSC9 CAN3 GPIO2	I/O I/O I I/O	VDD_IO	—	B18
I2C1_SDA	0x50 STD_PU_ST	ALT0 ALT1 ALT2 ALT3	I2C1_SDA PSC9_3 CAN3_TX GPIO50	I^2C1 PSC9 CAN3 GPIO2	I/O I/O O I/O	VDD_IO	—	A19
CAN1_RX	—	ALT0 ALT1 ALT2 ALT3	CAN1_RX — — —	CAN1 — — —	I — — —	VBAT	Dedicated input can be used to receive an external wakeup.	C9

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
CAN2_RX	—	ALT0 ALT1 ALT2 ALT3	CAN2_RX — — —	CAN2 — — —	I — — —	VBAT	Dedicated input can be used to receive an external wakeup.	B8
CAN1_TX	0x4D STD_PU_ST	ALT0 ALT1 ALT2 ALT3	CAN1_TX PSC9_0 I2C2_SCL GPIO47	CAN1 PSC9 I^2C2 GPIO2	O I/O I/O I/O	VDD_IO	—	B15
CAN2_TX	0x4E STD_PU_ST	ALT0 ALT1 ALT2 ALT3	CAN2_TX PSC9_1 I2C2_SDA GPIO48	CAN2 PSC9 I^2C2 GPIO2	O I/O I/O I/O	VDD_IO	—	A16
FEC1_TXD_2	0x51 STD_PU	ALT0 ALT1 ALT2 ALT3	FEC1_TXD_2 PSC2_0 USB2_DATA0 GPIO51	FEC1 PSC2 USB2 GPIO2	O I/O I/O I/O	VDD_IO	—	AA1
FEC1_TXD_3	0x52 STD_PU	ALT0 ALT1 ALT2 ALT3	FEC1_TXD_3 PSC2_1 USB2_DATA1 GPIO52	FEC1 PSC2 USB2 GPIO2	O I/O I/O I/O	VDD_IO	—	Y1
FEC1_RXD_2	0x53 STD_PU	ALT0 ALT1 ALT2 ALT3	FEC1_RXD_2 PSC2_2 USB2_DATA2 GPIO53	FEC1 PSC2 USB2 GPIO2	I I/O I/O I/O	VDD_IO	—	Y4
FEC1_RXD_3	0x54 STD_PU	ALT0 ALT1 ALT2 ALT3	FEC1_RXD_3 PSC2_3 USB2_DATA3 GPIO54	FEC1 PSC2 USB2 GPIO2	I I/O I/O I/O	VDD_IO	—	AA2
FEC1_CRS	0x55 STD_PU	ALT0 ALT1 ALT2 ALT3	FEC1_CRS PSC2_4 USB2_DATA4 GPIO55	FEC1 PSC2 USB2 GPIO2	I I/O I/O I/O	VDD_IO	—	U1

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
FEC1_TX_ER	0x56 STD_PU	ALT0 ALT1 ALT2 ALT3	FEC1_TX_ER PSC3_0 USB2_DATA5 GPIO56	FEC1 PSC3 USB2 GPIO2	O I/O I/O I/O	VDD_IO	—	W2
FEC1_RXD_1	0x57 STD_PU	ALT0 ALT1 ALT2 ALT3	FEC1_RXD_1/RMII_RX1 PSC3_1 USB2_DATA6 GPIO57	FEC1 PSC3 USB2 GPIO2	I I/O I/O I/O	VDD_IO	—	AA3
FEC1_TXD_1	0x58 STD_PU	ALT0 ALT1 ALT2 ALT3	FEC1_TXD_1/RMII_TX1 PSC3_2 USB2_DATA7 GPIO58	FEC1 PSC3 USB2 GPIO2	O I/O I/O I/O	VDD_IO	—	W3
FEC1_MDC	0x59 STD_PU	ALT0 ALT1 ALT2 ALT3	FEC1_MDC/RMII_MDC PSC3_3 USB2_DIR GPIO59	FEC1 PSC3 USB2 GPIO2	O I/O I I/O	VDD_IO	—	V1
FEC1_RX_ER	0x5A STD_PU	ALT0 ALT1 ALT2 ALT3	FEC1_RX_ER/RMII_RX_ER PSC3_4 USB2_NEXT GPIO60	FEC1 PSC3 USB2 GPIO2	I I/O I I/O	VDD_IO	—	Y5
FEC1_MDIO	0x5B STD_PU_ST	ALT0 ALT1 ALT2 ALT3	FEC1_MDIO/RMII_MDIO — USB2_CLK GPIO61	FEC1 — USB2 GPIO2	I/O — I I/O	VDD_IO	—	V2
FEC1_RXD_0	0x5C STD_PU	ALT0 ALT1 ALT2 ALT3	FEC1_RXD_0/RMII_RX0 — USB2_STOP GPIO62	FEC1 — USB2 GPIO2	I — O I/O	VDD_IO	—	AB2
FEC1_TXD_0	0x5D STD_PU_ST	ALT0 ALT1 ALT2 ALT3	FEC1_TXD_0/RMII_TX0 — NFC_R/$\overline{B}$1 GPIO63	FEC1 — NFC GPIO2	O — I I/O	VDD_IO	—	W4

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
FEC1_TX_CLK	0x5E STD_PU_ST	ALT0 ALT1 ALT2 ALT3	FEC1_TX_CLK/RMII_REF_CLK PSC0_0 — GPIO04	FEC1 PSC0 — GPIO1	I I/O — I/O	VDD_IO	—	W1
FEC1_RX_CLK	0x5F STD_PU_ST	ALT0 ALT1 ALT2 ALT3	FEC1_RX_CLK PSC0_1 NFC_R/$\overline{B}$2 GPIO05	FEC1 PSC0 — GPIO1	I I/O I I/O	VDD_IO	—	AB4
FEC1_RX_DV	0x60 STD_PU_ST	ALT0 ALT1 ALT2 ALT3	FEC1_RX_DV/RMII_CRS_DV PSC0_2 NFC_R/$\overline{B}$3 GPIO06	FEC1 PSC0 NFC GPIO1	I I/O I I/O	VDD_IO	—	AB3
FEC1_TX_EN	0x61 STD_PU	ALT0 ALT1 ALT2 ALT3	FEC1_TX_EN/RMII_TX_EN PSC0_3 — GPIO07	FEC1 PSC0 — GPIO1	O I/O — I/O	VDD_IO	—	Y3
FEC1_COL	0x62 STD_PU_ST	ALT0 ALT1 ALT2 ALT3	FEC1_COL PSC0_4 — GPIO08	FEC1 PSC0 GPIO1	I I/O — I/O	VDD_IO	—	U3
USB1_DATA0	0x63 STD_PU	ALT0 ALT1 ALT2 ALT3	USB1_DATA0 PSC1_0 FEC2_RXD_1/RMII_RX1 —	USB2 PSC1 FEC2	I/O I/O I —	VDD_IO	—	Y9
USB1_DATA1	0x64 STD_PU	ALT0 ALT1 ALT2 ALT3	USB1_DATA1 PSC1_1 FEC2_TXD_1/RMII_TX1 —	USB2 PSC1 FEC2	I/O I/O O —	VDD_IO	—	W9

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
USB1_DATA2	0x65 STD_PU	ALT0 ALT1 ALT2 ALT3	USB1_DATA2 PSC1_2 FEC2_MDC/RMII_MDC —	USB2 PSC1 FEC2	I/O I/O O —	VDD_IO	—	AB7
USB1_DATA3	0x66 STD_PU	ALT0 ALT1 ALT2 ALT3	USB1_DATA3 PSC1_3 FEC2_RX_ER/RMII_RX_ER —	USB2 PSC1 FEC2	I/O I/O I —	VDD_IO	—	AB6
USB1_DATA4	0x67 STD_PU	ALT0 ALT1 ALT2 ALT3	USB1_DATA4 PSC1_4 FEC2_MDIO/RMII_MDIO —	USB2 PSC1 FEC2	I/O I/O I/O —	VDD_IO	—	AA7
USB1_DATA5	0x68 STD_PU	ALT0 ALT1 ALT2 ALT3	USB1_DATA5 PSC4_0 FEC2_RXD_0/RMII_RX0 —	USB2 PSC4 FEC2	I/O I/O I —	VDD_IO	—	Y7
USB1_DATA6	0x69 STD_PU	ALT0 ALT1 ALT2 ALT3	USB1_DATA6 PSC4_1 FEC2_TXD_0/RMII_TX0 —	USB2 PSC4 FEC2	I/O I/O O —	VDD_IO	—	Y6
USB1_DATA7	0x6A STD_PU_ST	ALT0 ALT1 ALT2 ALT3	USB1_DATA7 PSC4_2 FEC2_TX_CLK/RMII_REF_CLK —	USB2 PSC4 FEC2	I/O I/O I —	VDD_IO	—	AB5
USB1_STOP	0x6B STD_PU_ST	ALT0 ALT1 ALT2 ALT3	USB1_STOP PSC4_3 FEC2_RX_CLK —	USB2 PSC4 FEC2	O I/O I —	VDD_IO	—	W6

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
USB1_CLK	0x6C STD_PU_ST	ALT0 ALT1 ALT2 ALT3	USB1_CLK PSC4_4 FEC2_RX_DV/RMII_CRS_DV —	USB2 PSC4 FEC2	I I/O I —	VDD_IO	—	Y8
USB1_NEXT	0x6D STD_PU	ALT0 ALT1 ALT2 ALT3	USB1_NEXT — FEC2_TX_EN/RMII_TX_EN GPIO09	USB2 — FEC2 GPIO1	I — O I/O	VDD_IO	—	AA5
USB1_DIR	0x6E STD_PU_ST	ALT0 ALT1 ALT2 ALT3	USB1_DIR — FEC2_COL GPIO10	USB2 — FEC2 GPIO1	I — I I/O	VDD_IO	—	W7
SDHC								
SDHC1_CLK	0x6F STD_PU	ALT0 ALT1 ALT2 ALT3	SDHC1_CLK NFC_CE1 FEC2_TXD_2 GPIO11	SDHC1 NFC FEC2 GPIO1	O O O I/O	VDD_IO	—	T1
SDHC1_CMD	0x70 STD_PU	ALT0 ALT1 ALT2 ALT3	SDHC1_CMD PSC5_0 FEC2_TXD_3 GPIO12	SDHC1 PSC5 FEC2 GPIO1	I/O I/O O I/O	VDD_IO	—	T2
SDHC1_D0	0x71 STD_PU	ALT0 ALT1 ALT2 ALT3	SDHC1_D0 PSC5_1 FEC2_RXD_2 GPIO13	SDHC1 PSC5 FEC2 GPIO1	I/O I/O I I/O	VDD_IO	—	T3
SDHC1_D1	0x72 STD_PU	ALT0 ALT1 ALT2 ALT3	SDHC1_D1_IRQ PSC5_2 FEC2_RXD_3 LPC_CS5	SDHC1 PSC5 FEC2 LPC	I/O I/O I O	VDD_IO	—	T4

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
SDHC1_D2	0x73 STD_PU	ALT0 ALT1 ALT2 ALT3	SDHC1_D2 PSC5_3 FEC2_CRS LPC_CS6	SDHC1 PSC5 FEC2 LPC	I/O I/O I O	VDD_IO	—	R1
SDHC1_D3	0x74 STD_PU	ALT0 ALT1 ALT2 ALT3	SDHC1_D3_CD PSC5_4 FEC2_TX_ER LPC_CS7	SDHC1 PSC5 FEC2 LPC	I/O I/O O O	VDD_IO	—	R2
PSC_MCLK_IN	0x75 STD_PU_ST	ALT0 ALT1 ALT2 ALT3	PSC_MCLK_IN — — GPIO14	 — — GPIO1	I — — I/O	VDD_IO	—	D6
PSC0_0	0x76 STD_PU	ALT0 ALT1 ALT2 ALT3	PSC0_0 SDHC2_CMD GPT1[0] GPIO15	PSC0 SDHC2 GPT1 GPIO1	I/O I/O I/O I/O	VDD_IO	—	C11
PSC0_1	0x77 STD_PU	ALT0 ALT1 ALT2 ALT3	PSC0_1 SDHC2_D0 GPT1[1] GPIO16	PSC0 SDHC2 GPT1 GPIO1	I/O I/O I/O I/O	VDD_IO	—	A12
PSC0_2	0x78 STD_PU	ALT0 ALT1 ALT2 ALT3	PSC0_2 SDHC2_D1_IRQ GPT1[2] GPIO17	PSC0 SDHC2 GPT1 GPIO1	I/O I/O I/O I/O	VDD_IO	—	A13
PSC0_3	0x79 STD_PU	ALT0 ALT1 ALT2 ALT3	PSC0_3 SDHC2_D2 GPT1[3] GPIO18	PSC0 SDHC2 GPT1 GPIO1	I/O I/O I/O I/O	VDD_IO	—	B13
PSC0_4	0x7A STD_PU	ALT0 ALT1 ALT2 ALT3	PSC0_4 SDHC2_D3_CD GPT1[4] CAN1_TX	PSC0 SDHC2 GPT1 CAN1	I/O I/O I/O O	VDD_IO	—	D11

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
PSC1_0	0x7B STD_PU	ALT0 ALT1 ALT2 ALT3	PSC1_0 SDHC2_CLK GPT1[5] CAN2_TX	PSC1 SDHC2 GPT1 CAN2	I/O O O O	VDD_IO	—	C12
PSC1_1	0x7C STD_PU	ALT0 ALT1 ALT2 ALT3	PSC1_1 CAN_CLK GPT1[6] IRQ0	PSC1 GPT1	I/O I/O I	VDD_IO	—	C13
PSC1_2	0x7D STD_PU	ALT0 ALT1 ALT2 ALT3	PSC1_2 TPA2 GPT1[7] IRQ1	PSC1 GPT1	I/O I/O I	VDD_IO	—	B14
PSC1_3	0x7E STD_PU	ALT0 ALT1 ALT2 ALT3	PSC1_3 CKSTP_IN NFC_R/$\overline{B}$2 GPIO19	PSC1 NFC GPIO1	I/O I I/O	VDD_IO	—	D13
PSC1_4	0x7F STD_PU	ALT0 ALT1 ALT2 ALT3	PSC1_4 CKSTP_OUT NFC_CE2 GPIO20	PSC1 MFC GPIO1	I/O O I/O	VDD_IO	—	A15
J1850_TX	0x80 STD_PU_ST	ALT0 ALT1 ALT2 ALT3	J1850_TX — NFC_CE3 I2C1_SCL	J1850 — NFC I^2C1	O — O I/O	VDD_IO	—	B5
J1850_RX	0x81 STD_PU_ST	ALT0 ALT1 ALT2 ALT3	J1850_RX — NFC_R/$\overline{B}$3 I2C1_SDA	J1850 — NFC I^2C1	I — I I/O	VDD_IO	—	C6
JTAG								
TCK	—	ALT0 ALT1 ALT2 ALT3	TCK — — —	JTAG — — —	I — — —	VDD_IO	5. This pin contains an enabled internal Schmitt trigger.	C16

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
TDI	—	ALT0 ALT1 ALT2 ALT3	TDI — — —	JTAG — — —	I — — —	VDD_IO	3. This JTAG pin has an internal pullup P-FET, and cannot be configured.	C15
TDO	—	ALT0 ALT1 ALT2 ALT3	TDO — — —	JTAG — — —	O — — —	VDD_IO	—	B16
TMS	—	ALT0 ALT1 ALT2 ALT3	TMS — — —	JTAG — — —	I — — —	VDD_IO	3. This JTAG pin has an internal pullup P-FET, and cannot be configured.	D15
$\overline{\text{TRST}}$	—	ALT0 ALT1 ALT2 ALT3	$\overline{\text{TRST}}$ — — —	JTAG — — —	I — — —	VDD_IO	3. This JTAG pin has an internal pullup P-FET, and cannot be configured.	D16
System Control								
$\overline{\text{HRESET}}$	—	ALT0 ALT1 ALT2 ALT3	$\overline{\text{HRESET}}$ — — —	 — — —	I — — —	VDD_IO	1. This pin is an input or open-drain output, and have internal pull-up P-FETs. This pin can not be configured. 5. This pin contains an enabled internal schmitt-trigger.	A17

Table 3-1. MPC5125 Pin Multiplexing (continued)

Pin	Pad I/O Control Register[1] and Offset[2]	Alternate Function[3]	Functions[4]	Peripheral[5]	I/O Direction	Power Domain	Notes	Pin
$\overline{\text{PORESET}}$	—	ALT0 ALT1 ALT2 ALT3	$\overline{\text{PORESET}}$ — — —	 — — —	I — — —	VDD_IO	1. This pin is an input or open-drain output, and have internal pull-up P-FETs. This pin can not be configured. 2. This pin is an input only. This pin cannot be configured. 5. This pin contains an enabled internal schmitt-trigger.	C17
$\overline{\text{SRESET}}$	—	ALT0 ALT1 ALT2 ALT3	$\overline{\text{SRESET}}$ — — —	 — — —	I — — —	VDD_IO	1. This pin is an input or open-drain output, and have internal pull-up P-FETs. This pin can not be configured. 5. This pin contains an enabled internal schmitt-trigger.	A18
Test/Debug								
TEST	—	ALT0 ALT1 ALT2 ALT3	TEST — — —	 — — —	I — — —	VDD_IO	2. This pin is an input only. This pin cannot be configured. 4. This test pin must be tied to VSS.	D14

1 Pins controlled by the STD_PU_ST register have a Schmitt trigger input; pins controlled by the STD_PU register do not. Pins controlled by the IO_CONTROL_MEM register access their alternate function ALT3 by setting the IO_CONTROL_MEM[16BIT] bit. This setting applies to all pins controlled by IO_CONTROL_MEM. Pins not controlled by these registers are indicated with a "—".

2 Offset from IOCONTROL_BASE (default is 0xFF40_A000).

3 Except where noted in the Notes column, ALT0 is the primary (default) function for each pin after reset.

4 Alternate functions are chosen by setting the values of the STD_PU[FUNCMUX] bitfields inside the I/O Control module.
– STD_PU[FUNCMUX] = 00 → ALT0 (default)
– STD_PU[FUNCMUX] = 01 → ALT1
– STD_PU[FUNCMUX] = 10 → ALT2
– STD_PU[FUNCMUX] = 11→ ALT3
For selecting alternate functions, the STD_PU and STD_PU_ST registers function the same. When no function is available on a pin's given ALT*n* function (value of STD_PU[FUNCMUX]), it is shown as "—".

5 Module included on the MCU.

3.2 Signal States During Reset

When a system reset is recognized ($\overline{\text{PORESET}}$ or $\overline{\text{HRESET}}$ are asserted), the MPC5125 aborts all current internal and external transactions, and it releases all bidirectional I/O signals to a high-impedance state. See Chapter 4, “Reset,” for a complete description of the reset functionality.

During reset, the MPC5125 ignores most input signals (except for reset configuration signals Table 3-2) and drives most of the output-only signals to an inactive state. Table 3-2, and Table 3-3 shows states of the output-only signals if boot from LPC or not boot from LPC.

Table 3-2. Signal States During System Reset (case boot from LPC)

Interface	Signal	State During Reset
MBA[2:0]	DDR bank select	All 0[1]
MA[15:0]	DDR address	16’h8000
$\overline{\text{MWE}}$	DDR write enable	1[1]
$\overline{\text{MRAS}}$	DDR row address strobe	1[1]
$\overline{\text{MCAS}}$	DDR column address strobe	1[1]
$\overline{\text{MCS}}$	DDR chip select	1[1]
MCKE	DDR clock enable	0
MCK	DDR differential clock	0
$\overline{\text{MCK}}$	DDR differential clock	1
MODT	DRAM On-Die Termination	0[1]
$\overline{\text{LPC\_OE}}$	LocalPlus output enable	1[1]
LPC_R/$\overline{\text{W}}$	LocalPlus read/write bar	1[1]
LPC_AX[3]	External Memory Bus address extension 3/LocalPlus Address latch	0[1]
EMB_AX[2]	External Memory Bus address extension 2/LocalPlus Address latch	0[1]
EMB_AX[1]	External Memory Bus address extension 1/LocalPlus Address latch	0[1]
EMB_AX[0]	External Memory Bus address extension 0/LocalPlus Address latch	0[1]
$\overline{\text{LPC\_ACK}}$	LPC Acknowledge	Pullup resistor
$\overline{\text{LPC\_CS0}}$	LPC Chip Select 0	Pullup resistor
$\overline{\text{NFC\_CE0}}$	NFC Chip Enable 0	Pullup resistor
DIU_LD00	CAN3_RX	Pullup resistor
DIU_LD01	CAN3_TX	Pullup resistor
DIU_LD08	CAN4_RX	Pullup resistor
DIU_LD09	CAN4_TX	Pullup resistor
DIU_LD16	I2C3_SCL	Pullup resistor
DIU_LD17	I2C3_SDA	Pullup resistor
I2C2_SCL	I2C2_SCL	Pullup resistor
I2C2_SDA	I2C2_SDA	Pullup resistor
I2C1_SCL	I2C1_SCL	Pullup resistor

Table 3-2. Signal States During System Reset (case boot from LPC) (continued)

Interface	Signal	State During Reset
I2C1_SDA	I2C1_SDA	Pullup resistor
CAN1_TX	CAN1_TX	Pullup resistor
CAN2_TX	CAN2_TX	Pullup resistor
PSC0_4	PSC0_4, in case function ALT3 (CAN1_TX) is used.	Pullup resistor
PSC1_0	PSC0_4, in case function ALT3 (CAN2_TX) is used.	Pullup resistor
SDHC1_CLK	SCHD clock	Pullup resistor
PSC1_4	PSC1_4	Pullup resistor
J1850_RX	J1850 Receive port	Pullup resistor
$\overline{\text{HRESET}}$	Hard reset	Pullup resistor
$\overline{\text{SRESET}}$	Soft reset	Pullup resistor
$\overline{\text{PORESET}}$	Power On Reset	Pullup resistor
Others		Z

Table 3-3. Output Signal States During System Reset (case NOT boot from LPC)

Interface	Signal	State During Reset
MBA[2:0]	DDR bank select	All 0[1]
MA[15:0]	DDR address	16'h8000
$\overline{\text{MWE}}$	DDR write enable	1[1]
$\overline{\text{MRAS}}$	DDR row address strobe	1[1]
$\overline{\text{MCAS}}$	DDR column address strobe	1[1]
$\overline{\text{MCS}}$	DDR chip select	1[1]
MCKE	DDR clock enable	0
MCK	DDR differential clock	0
$\overline{\text{MCK}}$	DDR differential clock	1
MODT	DRAM On-Die Termination	0[1]
LPC_AX[3]	External Memory Bus address extension 3/LocalPlus Address latch	Pullup resistor
EMB_AX[2]	External Memory Bus address extension 2/LocalPlus Address latch	Pullup resistor
EMB_AX[1]	External Memory Bus address extension 1/LocalPlus Address latch	Pullup resistor
$\overline{\text{LPC\_ACK}}$	LPC Acknowledge	Pullup resistor
$\overline{\text{LPC\_CS0}}$	LPC Chip Select 0	Pullup resistor
$\overline{\text{NFC\_CE0}}$	NFC Chip Enable 0	Pullup resistor
DIU_LD00	CAN3_RX	Pullup resistor
DIU_LD01	CAN3_TX	Pullup resistor
DIU_LD08	CAN4_RX	Pullup resistor
DIU_LD09	CAN4_TX	Pullup resistor
DIU_LD16	I2C3_SCL	Pullup resistor

Table 3-3. Output Signal States During System Reset (case NOT boot from LPC) (continued)

Interface	Signal	State During Reset
DIU_LD17	I2C3_SDA	Pullup resistor
I2C2_SCL	I2C2_SCL	Pullup resistor
I2C2_SDA	I2C2_SDA	Pullup resistor
I2C1_SCL	I2C1_SCL	Pullup resistor
I2C1_SDA	I2C1_SDA	Pullup resistor
CAN1_TX	CAN1_TX	Pullup resistor
CAN2_TX	CAN2_TX	Pullup resistor
PSC0_4	PSC0_4, in case function3 (CAN1_TX) is used.	Pullup resistor
PSC1_0	PSC0_4, in case function3 (CAN2_TX) is used.	Pullup resistor
SDHC1_CLK	SCHD clock	Pullup resistor
PSC1_4	PSC1_4	Pullup resistor
J1850_RX	J1850 Receive port	Pullup resistor
$\overline{\text{HRESET}}$	Hard reset	Pullup resistor
$\overline{\text{SRESET}}$	Soft reset	Pullup resistor
$\overline{\text{PORESET}}$	Power On Reset	Pullup resistor
Others		Z

The control device's signal multiplexing is documented in Chapter 20, "I/O Control."

This page is intentionally left blank.

Chapter 4
Reset

4.1 Introduction

The MPC5125 has three reset flows, $\overline{\text{PORESET}}$, $\overline{\text{HRESET}}$, and $\overline{\text{SRESET}}$. These flows can be initiated by the following events:

- Power on reset ($\overline{\text{PORESET}}$)
- Hard reset ($\overline{\text{HRESET}}$)
- Soft reset ($\overline{\text{SRESET}}$)
- JTAG reset command[1]
- Checkstop (MCP) event
- PLL unlock event
- Watchdog timer (WDT) module
- Bus monitor
- Software write to Reset Control Register (RCR)[2]

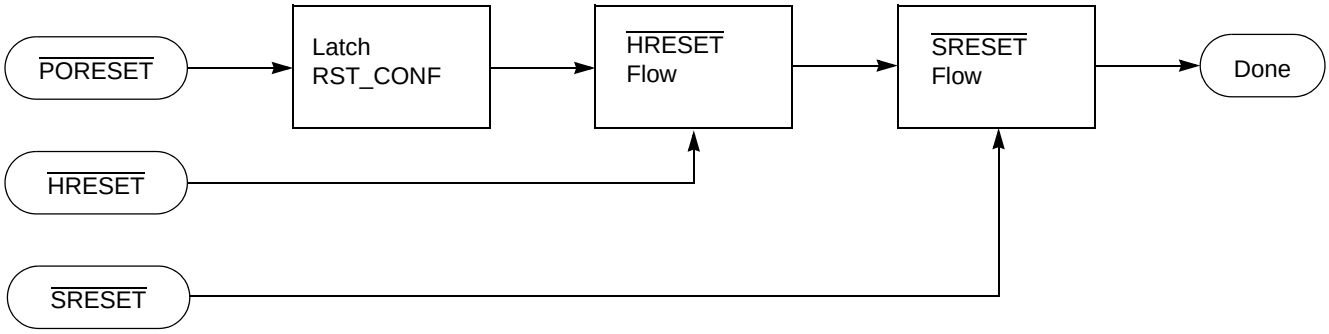


Figure 4-1. Internal Reset Flow

Table 4-1 summarizes each peripheral and the relation to each reset flow.

Table 4-1. Peripheral Versus Reset

Peripheral	$\overline{\text{HRESET}}$[1]	$\overline{\text{SRESET}}$
BDLC	—	x
CLOCK	x	—

1. The JTAG initiated reset (power on reset, hardware reset, and soft reset) is independent of the reset state of the JTAG controller ($\overline{\text{TRST}}$)
2. The Reset Control Register may initiate a $\overline{\text{SRESET}}$ or $\overline{\text{HRESET}}$ sequence. see Section 4.6.6, “Reset Control Register (RCR).”

Table 4-1. Peripheral Versus Reset (continued)

Peripheral	$\overline{\text{HRESET}}$[1]	$\overline{\text{SRESET}}$
CSBARB	—	x
DIU	—	x
DMA	—	x
e300[2]	x[2]	x[2]
EMB	—	x
FEC	—	x
FIFOC	—	x
FUSE	x	x
GPIO	x	—
GPT[3]	x[3]	x[3]
I2C	—	x
IIM	x	—
IO_CONTROL	x	—
IPIC	—	x
LPC	x	—
MDDRC	x	—
MEM	—	x
MEMMAP	x	—
MSCAN	—	x
NFC	—	x
PMC	—	x
PSC	—	x
RTC[4]	—	x
RESET	x	—
SDHC	—	x
TEMPSENS	—	—
TLM[5]	—	—
USB	—	x
WDT	—	x

1 $\overline{\text{PORESET}}$ causes the same effect as $\overline{\text{HRESET}}$.

2 e300 core supports two reset configurations. See the e300 core user's manual for details.

3 GPT supports two different reset configurations for each reset type.

4 RTC is reset by V_{BAT} Power-On-Reset

5 TLM can only be reset through use of the JTAG (TRST). See Section 4.5.3, "JTAG Reset."

4.2 Power-On Initialization ($\overline{\text{PORESET}}$)

During the power-up sequence the $\overline{\text{PORESET}}$ pin must be asserted by an external device for a minimum of 32 XTAL clock cycles. The oscillator input (XTALI) must be stable prior to $\overline{\text{PORESET}}$ de-assertion. The reset configuration word is latched on the de-assertion of Power On Reset. After $\overline{\text{PORESET}}$ has been qualified, the $\overline{\text{HRESET}}$ flow is started.

4.3 $\overline{\text{HRESET}}$ Flow

$\overline{\text{HRESET}}$ provides a mechanism to initialize all clocks and peripherals to their initial values. The CPMF and SPMF bits, which set the feedback ratio for the Core and System PLLs, are not affected by $\overline{\text{HRESET}}$. The reset configuration word is sampled only when the $\overline{\text{PORESET}}$ is deasserted.

4.3.1 Sources

The following sources may initiate an $\overline{\text{HRESET}}$ sequence:

- $\overline{\text{PORESET}}$ input signal
- $\overline{\text{HRESET}}$ input signal
- Watchdog timer (WDT) module
- JTAG command
- Bus monitor
- Checkstop event[1]
- PLL unlock event
- Software write to the RESET module[2]

4.3.2 Impacts

When a $\overline{\text{HRESET}}$ sequence is initiated, the following occurs:

- $\overline{\text{HRESET}}$ pin is asserted by the device
- $\overline{\text{SRESET}}$ pin is asserted by the device.
- MSR[IP] bit in the e300 core is updated to reflect the vector table location
- PLLs reload programming information requiring the PLL re-lock to the reference clock signal
- Clock dividers are initialized
- Memory map initializes to reset state
- All peripheral logic asserts reset unless otherwise noted[3]
- Reset source is captured in the RESET module
- The e300 core starts fetching instructions from the vector indicated by the RST_CONF word

1. Checkstop may be initiated when the e300 core enters the checkstop state. This state may be masked in the RESET module, e300 core or IPIC.
2. The reset configuration register in the RESET module may initiate either a $\overline{\text{SRESET}}$ or $\overline{\text{HRESET}}$ sequence.
3. The RTC timer mechanism retains state as long as V_{bat} provides power to the RTC module.

- Real time clock shadow registers and time, date, alarm, and stopwatch registers initialize to reset state. Other RTC registers are not reset.

4.4 $\overline{\text{SRESET}}$ Flow

$\overline{\text{SRESET}}$ provides a mechanism to shorten the boot flow by bypassing initialization of boot peripherals and clocks (see Table 4-1). Timing for $\overline{\text{SRESET}}$ can be found in the *MPC5125 Microcontroller Data Sheet*.

4.4.1 Sources

The following sources initiate an $\overline{\text{SRESET}}$ sequence:

- $\overline{\text{SRESET}}$ input signal
- JTAG JSRS command
- Software write to the RESET module

4.4.2 Impacts

When $\overline{\text{SRESET}}$ sequence is initiated, the following occurs:

- $\overline{\text{SRESET}}$ pin is asserted by the device
- The following peripherals are not affected by reset:
  - IO control and pin multiplexing
  - Clock control registers
  - Memory access windows (XLBMEN)
  - Local plus memory controller (LPC)
  - DRAM controller (MDDRC)
  - RTC registers on the V_{BAT} power domain (see Section 4.5.2, “RTC at Reset”)
- Reset source is captured in the RESET module
- MSR[IP] bit in the e300 core is not updated
- The e300 core starts execution at the reset vector
- See e300 core manual for core impact of $\overline{\text{SRESET}}$

4.5 Reset Configuration Word (RST_CONF)

The RST_CONF word is latched 3 oscillator clock cycles after $\overline{\text{PORESET}}$ is deasserted. This controls the boot configuration of the device. Each RST_CONF pin MUST have external pull-up/pull-down devices that ensure that the device enters the desired mode of operation. The value latched into the device at reset may be verified by access to the RCWLR and RCWHR registers (see Section 4.6, “Memory Map”).

Available modes include:

- Test modes
- Boot interface selection

- MUX Flash mode
- NOR Flash port size
- Core PLL programming
- System PLL programming
- Clock divider

Table 4-2 lists the parameters in the RST_CONF word and the pins associated with those parameters.

Table 4-2. Reset Configuration Word

Reset Parameter	Signal	Description
RST_CONF_BMS	EMB_AD[2]	Boot mode select — Selects e300 boot vector[1] and configures default value for LPC CS0 or NFC base address. See Section 4.5.1, "BMS Operation."
RST_CONF_ROM_LOC	EMB_AD[1:0]	Selects boot device. 00 LPC boot. 01 NAND (NFC) boot[2]. 10 Reserved. 11 Reserved.
RST_CONF_LPC_DBW	EMB_AD[4:3]	LPC Data Port Size. 00 8-bit. 01 16- bit. 10 Reserved. 11 32-bit.
RST_CONF_COREPLL	EMB_AD[7:5]	Core PLL Multiply factor. See Section 5.2.8.2, "e300 Core PLL Programming Model," for programming options.
RST_CONF_SYSPLL	EMB_AD[11:8]	System PLL Multiply factor. See Section 5.2.8.1, "System PLL Programming Model," for programming options.
RST_CONF_SYSOSCEN	EMB_AD[15]	Oscillator Bypass Mode. 0 System Oscillator bypass mode. 1 System Oscillator mode.
RST_CONF_SYSDIV	EMB_AD[14:12]	System PLL divider ratio. See clock module for programming options.
Reserved	EMB_AD[16]	Reserved. Must be connected to 1.
Reserved	EMB_AD[17]	Reserved. Must be connected to 0.
RST_CONF_LPCMX	EMB_AD[18]	LPC MUXED mode. 0 Non-muxed mode. 1 Muxed mode.
RST_CONF_LPCWA	EMB_AD[19]	LPC Word/Byte address. 0 Address is interpreted as byte address. 1 Address is interpreted as word address.
RST_CONF_LPC_TS	EMB_AD[22]	Use LPC_TS, LPC_TSIZ0, LPC_TSIZ1 if boot from LPC. 0 Disable LPC_TS, LP_TSIZ0, LPC_TSIZ1 output if boot from LPC. 1 Enable LPC_TS, LPC_TSIZ0, LPC_TSIZ1 output if boot from LPC.

1 Only valid when LPC boot mode is selected (ROM_LOC).

2 NFC memory access windows are placed at the location indicated by the BMS bit.

4.5.1 BMS Operation

The Boot Mode Select (BMS) bit determines the default value of LPC CS0 and provides the reset vector to the e300 core. The e300 MSR[IP] bit reflects the state which is latched by the BMS bit. The BMS bit indicates to the e300 where in memory to fetch the first instruction.

Table 4-3. BMS Impact on Boot Vector

Parameter	BMS = 0 (boot low)	BMS = 1 (boot high)
e300 Boot Vector	0x0000_0100	0xFFF0_0100

Table 4-4. BMS Impact On Memory Windows

Parameter	BMS = 0 (boot low)	BMS = 1 (boot high)
LPC CSBOOT Start	0x0000_0000	0xFF80_0000
LPC CSBOOT End	0x007F_FFFF	0xFFFF_FFFF
NFC Base Address[1]	0x0000_0000	0xFFF0_0000

1 Valid only when ROM_LOC indicates NAND Flash Boot.

4.5.2 RTC at Reset

The RTC module contains registers located on the V_{BAT} power domain that are not affected by system level reset functions. These registers can only be reset by removing power from V_{BAT}.

4.5.3 JTAG Reset

The JTAG state machine is reset by the assertion of the JTAG $\overline{\text{TRST}}$ pin. The JTAG circuitry is not affected by the assertion of $\overline{\text{PORESET}}$, $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$. Even if your system does not utilize a JTAG connector, the JTAG $\overline{\text{TRST}}$ and TCLK pins must be tied to a defined state or undefined behavior may occur.

4.5.4 Boot Vector Selection

The e300 boot vector may be configured through use of the RST_CONF_ROM_LOC and RST_CONF_BMS pins at reset (see Section 4.5, “Reset Configuration Word (RST_CONF)”). These pins allow selection of the boot memory interface and/or e300 boot vector.

4.5.5 Boot Memory Interface Selection

The e300 boot memory interface is selected by configuring the RST_CONF_ROM_LOC pin at reset. This allows selection of either the local plus controller (LPC) or NAND flash controller as the boot memory device. See Section 4.5.1, “BMS Operation,” for details. Each interface requires a unique boot strap sequence for initializing the MPC5125. These initialization sequences are described in the following sections.

4.5.6 LPC Initialization Sequence

This interface is utilized when the ROM_LOC configuration in the RST_CONF selects the Local Plus Interface as the boot vector. The following boot sequence works with both boot high or boot low vectors.

Table 4-5. LPC Initialization Sequence

Step		Software Region	Note
1	Configure IMMR[1]	Reset Vector (Flash)	—
2	Configure LPC Clock Dividers	Reset Vector (Flash)	—
3	Configure CS0 Access Window	Reset Vector (Flash)	—
4	Configure CS0 Timing parameters	Reset Vector (Flash)	—
5	Perform an absolute jump to the initialization routine	Reset Vector (Flash)	Relative branching should not be used
6	Initialize Memory Map	Startup (Flash)	Initialize all memory access windows and LPC chip selects
7	Initialize e300 Core	Startup (Flash)	Initialize core settings including cache policies and instruction burst capabilities. Flash should be initialized as both I/D cached in copyback mode[2].
8	Initialize system clocks	Startup (Flash)	—
9	Initialize DRAM	Startup (Flash)	—
10	Initialize IO pin muxing	Startup (Flash)	—
11	Code relocation	Startup (Flash)	Relocate code into DRAM for faster execution
12	c-runtime initialization	Startup (Flash)	This allows remaining routines
13	Peripheral initialization	Startup (Flash)	Perform any initialization required by monitor or RTOS here.
14	Boot Into Application Space	DRAM	Start Application Environment (RTOS)

1 Care must be taken to ensure that the IMMR address region does not overlap any active access windows at any time during the boot process.

2 This MUST be changed to cache inhibited prior to entry into user code space.

4.5.7 NFC Initialization Sequence

This interface is utilized when the ROM_LOC configuration in the RST_CONF selects the NFC (NAND Flash) Interface as the boot device. For more information see Chapter 23, “NAND Flash Controller (NFC).”

Table 4-6. NFC Initialization Sequence

Step		Software Region	Note
1	Configure IMMR[1]	Reset Vector (Flash)	—
2	Configure DRAM & NFC Clock Dividers	Reset Vector (Flash)	—
3	Configure NFC parameters	Reset Vector (Flash)	—

Table 4-6. NFC Initialization Sequence (continued)

Step		Software Region	Note
4	Initialize DRAM	Reset Vector (Flash)	Initialization should include DRAM access window as well as timings and initialization
5	Copy NFC bootstrap to DRAM	Reset Vector (Flash)	—
6	Perform absolute jump to NFC bootstrap	NFC Bootstrap (DRAM)	Relative branch should not be used.
7	Initialize Memory Map	Startup (DRAM)	Initialize all memory access windows and LPC chip selects
8	Code re-location	Startup (DRAM)	Relocate code into DRAM for faster execution. Must use nand flash mini-driver to access NAND flash
9	Initialize e300 Core	Startup (DRAM)	Initialize core settings including cache policies and instruction burst capabilities. Flash should be initialized as both I/D cached in copyback mode[2].
10	Initialize system clocks	Startup (DRAM)	—
11	Initialize IO pin muxing	Startup (DRAM)	—
12	c-runtime initialization	Startup (DRAM)	This allows remaining routines
13	peripheral initialization	Startup (DRAM)	Perform any initialization required by monitor or RTOS here.
14	Boot Into Application Space	(DRAM)	Start Application Environment (RTOS)

1 Care must be taken to ensure that the IMMR address region does not overlap any active access windows at any time during the boot process.

2 This MUST be changed to cache inhibited prior to entry into user code space.

4.6 Memory Map

The reset configuration and status registers are shown in Table 4-7.

Table 4-7. Reset Configuration Registers Memory Map

Offset from RESET_BASE (0xFF40_0E00)[1]	Register	Access	Reset Value[2]	Section/Page
0x00	Reset Configuration Word Low (RCWL) Register	R	0x0U0U_0U00	4.6.1/4-69
0x04	Reset Configuration Word High (RCWH) Register	R	0xUUU0_UU00	4.6.2/4-69
0x08–0x0F	Reserved			
0x10	Reset Status Register (RSR)	R/W	0x6000_0000	4.6.3/4-70
0x14	Reset Mode Register (RMR)	R/W	0x0000_0000	4.6.4/4-72
0x18	Reset Protection Register (RPR)	R/W	0x0000_0000	4.6.5/4-73
0x1C	Reset Control Register (RCR)	R/W	0x0000_0000	4.6.6/4-73
0x20	Reset Control Enable Register (RCER)	R/W	0x0000_0000	4.6.7/4-74
0x24–0xFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 In this column, the symbol “U” indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

4.6.1 Reset Configuration Word Low (RCWL) Register

The Reset Configuration Word Low register is shown in Figure 4-2. This read-only register gets its value according to the reset configuration word low loaded during the reset flow.

Address: Base + 0x00 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SPMF				0	0	0	0	0	CPMF		
W																
Reset	0	0	0	0	EMB_AD11[1]	EMB_AD10[1]	EMB_AD9[1]	EMB_AD8[1]	0	0	0	0	0	EMB_AD7[1]	EMB_AD6[1]	EMB_AD5[1]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	SYSDIV			0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	EMB_AD14[1]	EMB_AD13[1]	EMB_AD12[1]	0	0	0	0	0	0	0	0

Figure 4-2. Reset Configuration Word Low (RCWL) Register

1 See Table 4-2. for more information.

Table 4-8. RCWL field descriptions

Field	Description
SPMF	System PLL multiplication factor. See Section 5.2.8.1, “System PLL Programming Model.”
CPMF	Core PLL configuration. See Section 5.2.8.2, “e300 Core PLL Programming Model.”
SYSDIV	System clock divide factor. See Section 5.3.1.5, “System Clock Frequency Register 2 (SCFR2).”

4.6.2 Reset Configuration Word High (RCWH) Register

The Reset Configuration Word High register is shown in Figure 4-3. This read-only register gets its value according to the reset configuration word high loaded during the reset flow.

Address: Base + 0x04 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		0	0	SYS OSC EN	0	BMS	0	0	0	ROM_LOC		0	0	0	0	0
W																
Reset	EMB AD16[1]	0	0	EMB AD15[1]	0	EMB_ AD02[1]	EMB AD17[1]	0	0	EMB_ AD01[1]	EMB_ AD00[1]	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	LPC_ TS	LPC_ MX	LPC_ WA	LPC_DBW		0	0	0	0	0	0	0	0
W																
Reset	0	0	0	EMB_ AD22[1]	EMB_ AD18[1]	EMB_ AD19[1]	EMB_ AD04[1]	EMB_ AD03[1]	0	0	0	0	0	0	0	0

Figure 4-3. Reset Configuration Word High (RCWH) Register

1 See Table 4-2. for more information.

Table 4-9. RCWH field descriptions

Field	Description
SYSOSCEN	System Oscillator Enable. Reset value is read from the EMBAD15 pin at reset.
BMS	Boot mode select. Reset value is read from the EMBAD02 pin at reset. See Section 4.5.1, “BMS Operation.”
ROM_LOC[1:0]	Boot ROM interface location
LPC_TS	LPC_TS mode
LPC_MX	LPC MUXED mode
LPC_WA	LPC word/byte address
LPC_DBW[1:0]	LPC data bus width

NOTE

The value of fields in the reset configuration words registers (RCWLR and RCWHR) reflect only their state during the reset flow. Some of these parameters and modes may be modified by changing their values in other units’ memory mapped registers. Modifying values in these other units’ memory mapped registers do not affect RCWLR and RCWHR.

4.6.3 Reset Status Register (RSR)

The Reset Status Register (RSR) shown in Figure 4-4 captures various reset events in the device.

Address: Base + 0x10 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	SWSR	SWHR	0	JPRS	JHRS	JSRS	PURS	EXT1HRS	0	CSHR	SWRS	BMRS	SRS	HRS
W			w1c	w1c		w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-4. Reset Status Register (RSR)

Table 4-10. RSR field descriptions

Field	Description
SWSR	Software soft reset. If set, indicates that a software soft reset has occurred. SWSR is cleared by writing a logic 1 to it.
SWHR	Software hard reset. If set, indicates that a software hard reset has occurred. SWHR is cleared by writing a logic 1 to it.
JPRS	JTAG POReset status. When the JTAG reset (command) request is set, JPRS is set and remains set until software clears it. JPRS is cleared by writing a logic 1 to it. 0 No JTAG reset event occurred. 1 A JTAG reset event occurred.
JHRS	JTAG HReset status. When the JTAG reset (command) request is set, JHRS is set and remains set until software clears it. JHRS is cleared by writing a logic 1 to it. 0 No JTAG reset event occurred. 1 A JTAG reset event occurred.
JSRS	JTAG SReset status. When the JTAG reset (command) request is set, JSRS is set and remains set until software clears it. JSRS is cleared by writing a logic 1 to it. 0 No JTAG reset event occurred. 1 A JTAG reset event occurred.
PURS	PLL Unlock reset status. PURS is cleared by writing a logic 1 to it. 0 No System PLL unlock event has occurred. 1 System PLL unlock event has occurred.
EXT1HRS	External HRESET1 status. EXT1HRS is cleared by writing a logic 1 to it. 0 No external HRESET1 event has occurred. 1 External HRESET1 event has occurred.
CSHR	Check stop reset status. When the core enters a checkstop state and the checkstop reset is enabled by the RMR[CSRE]. CSHR is cleared by writing a logic 1 to it. 0 No enabled check stop reset event occurred. 1 An enabled check stop reset event occurred.
SWRS	Software watchdog reset status. When a software watchdog expire event (which causes a reset) is detected, the SWRS bit is set and remains that way until the software clears it. SWRS is cleared by writing a logic 1 to it. 0 No software watchdog reset event occurred. 1 A software watchdog reset event has occurred.

Table 4-10. RSR field descriptions (continued)

Field	Description
BMRS	Bus monitor reset status. When a bus monitor expire event (which causes a reset) is detected, BMRS is set and remains set until the software clears it. BMRS is cleared by writing a logic 1 to it. 0 No bus monitor reset event has occurred. 1 A bus monitor reset event has occurred.
SRS	Soft reset status. When an external or internal soft reset event is detected, SRS is set and remains that way until software clears it. SRS is cleared by writing a logic 1 to it. 0 No soft reset event has occurred. 1 A soft reset event has occurred. **Note:** Soft reset induced by hard reset also sets this bit.
HRS	Hard reset status. When an external or internal hard reset event is detected, HRS is set and remains that way until software clears it. HRS is cleared by writing a logic 1 to it. 0 No hard reset event has occurred. 1 A hard reset event has occurred.

NOTE

The Reset Status Register accumulates reset events. This register returns to its reset value only when power-on reset occurs.

4.6.4 Reset Mode Register (RMR)

The Reset Mode Register (RMR), shown in Figure 4-5, controls the internal reset sources.

Address: Base + 0x14 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	DP KILL	0	0	PURE	CSRE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-5. Reset Mode Register (RMR)

Table 4-11. RMR field descriptions

Field	Description
DPKILL	Disable PLL Kill 0 Normal operation, during HRESET phase the system PLL is disabled to force a relock of the system PLL. 1 During HRESET phase the system PLL is still enabled; no relocking is required. The system PLL will be disabled if the PURE was the source of the reset.
PURE	PLL unlock reset enable 0 Reset is not generated when the system PLL unlocks. 1 Reset generated when the system PLL unlocks.

Table 4-11. RMR field descriptions (continued)

Field	Description
CSRE	Checkstop reset enable. The e300 core can enter checkstop mode as the result of several exception conditions. Setting CSRE configures the chip to perform a hard reset sequence whenever the e300 core enters checkstop state. 0 Reset is not generated when the core enters checkstop state. 1 Reset is generated when the core enters checkstop state.

4.6.5 Reset Protection Register (RPR)

The Reset Protection Register (RPR), shown in Figure 4-6, enables or disables writing to the Reset Control Register (RCR). This register prevents unwanted resets due to unintended software writes to the Reset Control Register (RCR). The user should write the value 0x5253_5445 (RSTE in ASCII) to enable. Enable indication appears in the Reset Control Enable Register (RCER[CRE]). Reading this register always returns all 0s. To disable writing to the Reset Control Register (RCR), write a 1 to RCER[CRE].

Address: Base + 0x18 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RCPW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RCPW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-6. Reset Protection Register (RPR)

Table 4-12. RPR field descriptions

Field	Description
RCPW	Reset control protection word. This register prevents software reset requests from occurring because of unintended writes to the Reset Control Register (RCR). To enable write protection, write 0x5253_5445 (RSTE in ASCII) to this bitfield. Enable indication appears in the Reset Control Enable Register (RCER[CRE]). Reading this register always returns all 0s. To disable write to the RCR, write 1 to RCER[CRE].

4.6.6 Reset Control Register (RCR)

The Reset Control Register (RCR) shown in Figure 4-7 can be used by software to initiate a soft or hard reset sequence. To allow writing to this register, the user must first write the value 0x5253_5445 to the Reset Protection Register (RPR).

Address: Base + 0x1C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SW HR	SW SR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-7. Reset Control Register (RCR)

Table 4-13. RCR field descriptions

Field	Description
SWHR	Software hard reset. Setting this bit causes the MPC5125 to begin a hard reset flow. This bit returns to its reset state during the reset sequence, so reading it always returns 0.
SWSR	Software soft reset. Setting this bit causes the MPC5125 to begin a soft reset flow. This bit returns to its reset state during the reset sequence, so reading it always returns 0.

4.6.7 Reset Control Enable Register (RCER)

The Reset Control Enable Register (RCER), shown in Figure 4-8, indicates by the CRE field that the Reset Protection Register (RPR) was accessed with a value that enables the Reset Control Register (RCR).

Address: Base + 0x20 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CRE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-8. Reset Control Enable Register (RCER)

Table 4-14. RCER field descriptions

Field	Description
CRE	Control register enabled. When set, indicates that the Reset Protection Register (RPR) was accessed with a value that enables the Reset Control Register (RCR). Writing 1 to this bit disables the Reset Control Register (RCR) and then clears the CRE bit. Writing 0 has no effect.

Chapter 5
Clocks and Low-Power Modes

5.1 Introduction

The wide range of applications supported by the MPC5125 requires a complex clocking structure with different primary clock domains derived from an oscillator source. Internal PLL and clock dividers allow generation of a wide range of clock references.

Each peripheral clock may be individually controlled to minimize total power consumption of the device. Clocks may be gated individually or scaled in frequency to ensure the most efficient power profile for the user's application.

5.2 System Clock Generation

The system reference is provided by a crystal oscillator that drives the system PLL. This PLL is programmed at reset by the reset configuration word (RST_CONF) sampled at the rising edge (deassertion) of power on reset ($\overline{\text{PORESET}}$). See Section 4.2, "Power-On Initialization (PORESET)." The SYS_PLL clock is then divided (SYS_DIV) and used as a reference to the MPC5125 cores and peripherals.

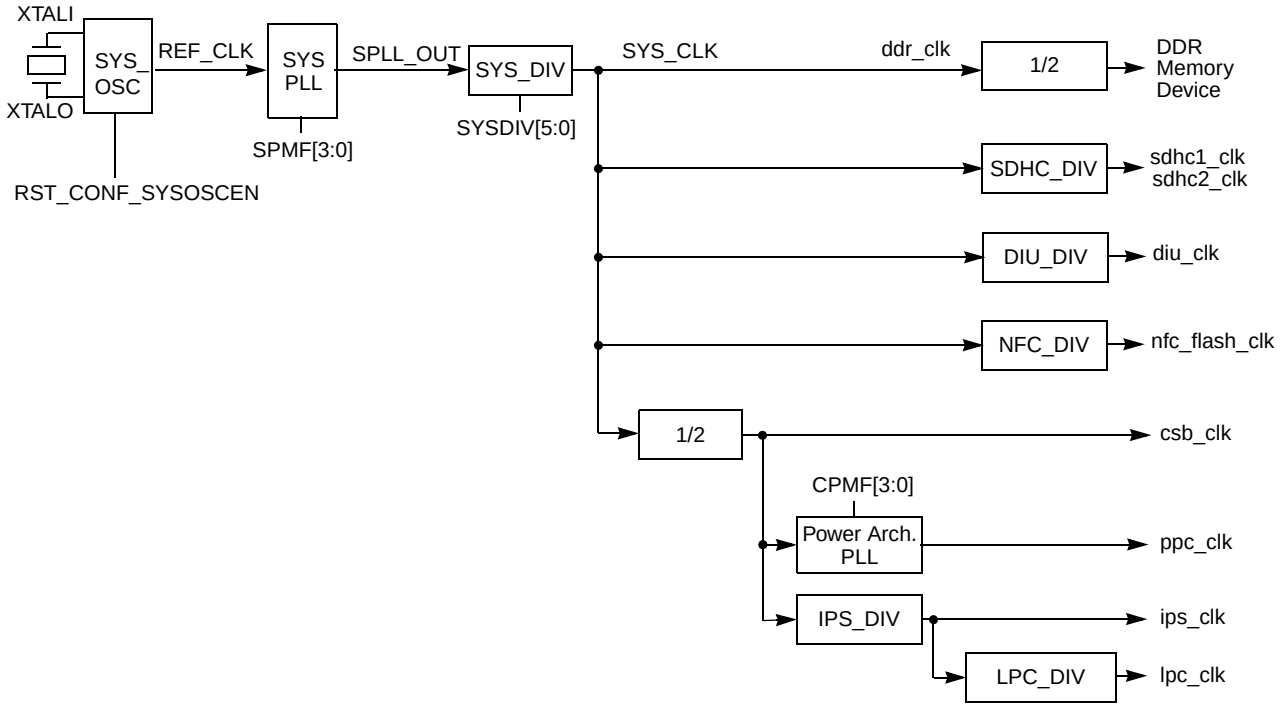


Figure 5-1. MPC5125 Clock System

Table 5-1. Clock Domain Programming

Domain	Programming Interface
SYS_PLL	Hardware Programmable (Section 5.2.8.1, “System PLL Programming Model”)
CORE_PLL	Hardware Programmable (Section 5.2.8.2, “e300 Core PLL Programming Model”)
IPS_DIV	Software Programmable (Section 5.3.1, “Memory Map/Register Definition”)
NFC_DIV	Software Programmable (Section 5.3.1, “Memory Map/Register Definition”)
LPC_DIV	Software Programmable (Section 5.3.1, “Memory Map/Register Definition”)
DIU_DIV	Software Programmable (Section 5.3.1, “Memory Map/Register Definition”)
PSC_DIV[1]	Software Programmable (Section 5.3.1, “Memory Map/Register Definition”)
MSCAN_DIV[2]	Software Programmable (Section 5.3.1, “Memory Map/Register Definition”)

1 PSC clock generation sub-system is described in Section 5.2.3, “PSC Clock Generation,” on page 5-77.

2 MSCAN clock generation sub-system is described in Section 5.2.4, “MSCAN Clock Generation,” on page 5-78.

5.2.1 Peripheral Clock Domains

Table 5-2 lists the reference clock for each peripheral.

Table 5-2. Peripheral Clock Reference

Peripheral	Reference Clock
BDLC	ips_clk
CSBARB	csb_clk
DIU	diu_clk, csb_clk, ips_clk
DMA	csb_clk
E300	ppc_clk
PPC_PLL	csb_clk
EMB	ips_clk, lpc_clk, nfc_clk, csb_clk
FEC1, FEC2	ips_clk
FIFOC	ips_clk
FUSE	ips_clk
GPIO1, GPIO2	ips_clk
I2C	ips_clk
IIM	ips_clk
IPIC	ips_clk
LPC	lpc_clk
MDDRC	ddr_clk
MEM	csb_clk
MSCAN	ips_clk[1]
NFC	nfc_clk, csb_clk
PMC	ips_clk, REF_CLK
PRIMAN	csb_clk

Table 5-2. Peripheral Clock Reference (continued)

Peripheral	Reference Clock
PSC[0:9]	ips_clk[2]
RTC	ips_clk, rtc_clk
SDHC1, SDHC2	ips_clk, sdhc_clk
SYS_PLL	ref_clk
TLM	tck
USB1, USB2	ips_clk, usb_clk
WDT	ips_clk, ref_clk

1 For all clock sources, see Section 5.2.4, "MSCAN Clock Generation."
2 For all clock sources, see Section 5.2.3, "PSC Clock Generation."

Many peripheral clocks may be disabled to reduce power consumption. See Section 5.3.1.2, "System Clock Control Register 1 (SCCR1)," for more information.

5.2.2 System Oscillator Disable

The system oscillator can be disabled to allow an externally generated clock to be used as a reference. This can be performed by setting the RST_CONF_SYSOSCEN in the reset configuration word (RST_CONF) at reset.

5.2.3 PSC Clock Generation

Each PSC can select from multiple clock sources. A single clock input is provided that allows the PSC_MCLK_IN to be used as a master reference by all PSCs. Additionally, clock gating is supplied allowing the shutdown of unnecessary clocks.

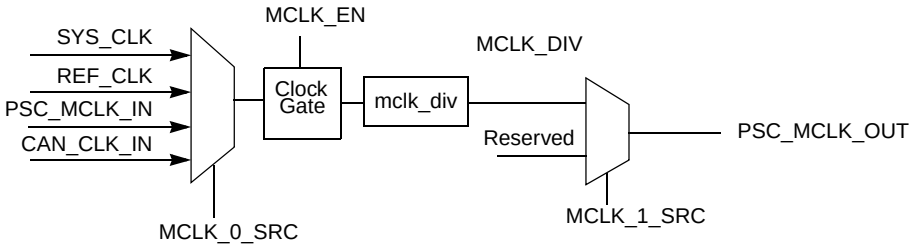


Figure 5-2. PSC (MCLK) Clock Generation[1]

This circuit is replicated for each PSC and can be controlled by accessing the PSC Clock Control Registers (see Section 5.3.1.8, "PSC0 Clock Control Register (P0CCR)," and subsequent sections).

1. PSC_MCLK_IN and CAN_CLK_IN are generated by external pins.

5.2.4 MSCAN Clock Generation

Each MSCAN module can select from multiple clock sources. A single clock input is provided that allows the PSC_MCLK_IN to be used as a clock source for all MSCAN modules. Additionally, clock gating is supplied allowing the shutdown of unnecessary clocks.

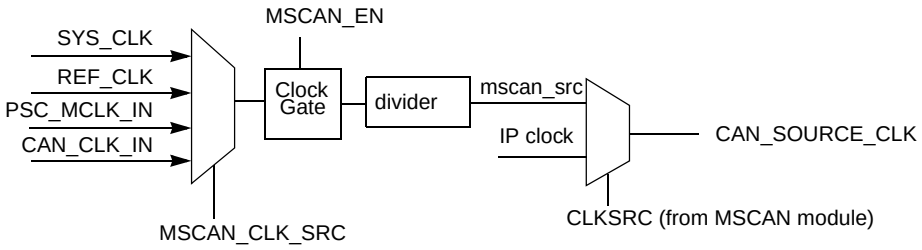


Figure 5-3. MSCAN Source Clock Generation[1]

This circuit is replicated for each MSCAN module and can be controlled by accessing the MSCAN Clock Control Registers (see Section 5.3.1.19, “MSCAN1 Clock Control Register (M1CCR),” and subsequent sections).

5.2.5 OUT Clock Generation

Internal clock mux logic provides selectable clock sources.

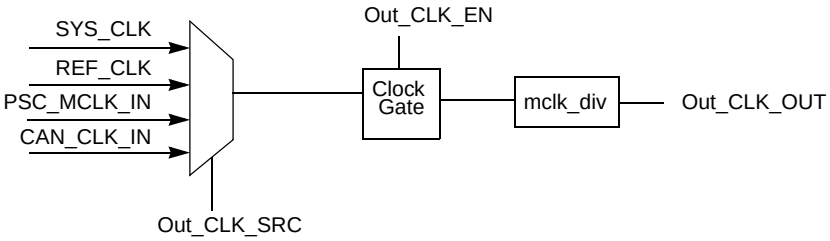


Figure 5-4. Out Clock Generation

5.2.6 RTC Clock Generation

The RTC module contains circuitry on two clock and voltage domains. The V_{BAT} voltage domain circuitry operates from a 32.768 kHz oscillator input. The programming interface operates from the clock reference provided by the IP_BUS interface.

1. PSC_MCLK_IN and CAN_CLK_IN are generated by external pins.

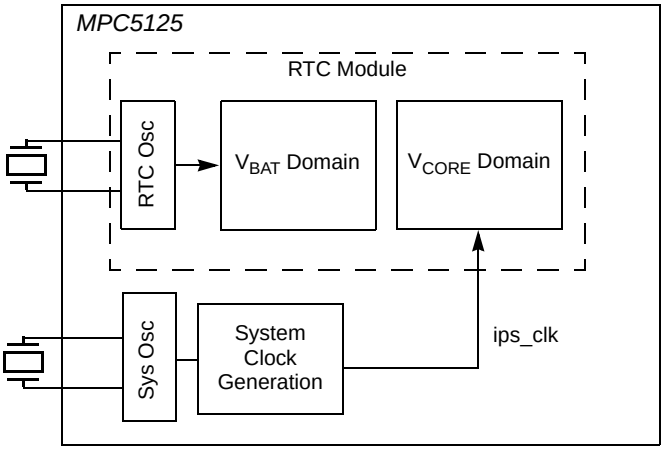


Figure 5-5. RTC Clock Generation

5.2.7 System PLL lock detection

The clock system provides a logic to observe the System PLL output frequency. A deviation of more than 1% of the target frequency of the PLL output clock (spll_out) will be indicated as unlock. The time between the $\overline{\text{PORESET}}$ and the first lock event will be stored in the SYSPLL_LOCK_CNT register. For more information, see Section 5.3.1.28, "System PLL Lock Counter Register (SPLL_LOCK_CNT)."

5.2.8 System PLL and e300 PLL

5.2.8.1 System PLL Programming Model

The output of the system PLL conforms to the following equation:

$$f_{spll} = SPMF \times f_{ref\_clk} \qquad \textit{Eqn. 5-1}$$

Table 5-3. System PLL Programming—Normal Mode

Field Name	Value (Binary)	$f_{spll} : f_{ref\_clk}$
SPMF	0000	68:1
	0001	PLL BYPASS[1]
	0010	12:1
	0011	16:1
	0100	20:1
	0101	24:1
	0110	28:1
	0111	32:1
	1000	36:1
	1001	40:1
	1010	44:1
	1011	48:1
	1100	52:1
	1101	56:1
	1110	60:1
	1111	64:1

1 All the clocks from CLOCK block are bypassed with PLL reference clock.

5.2.8.2 e300 Core PLL Programming Model

Table 5-4. Core PLL Programming—Normal Mode

Field Name	Value (Binary)	$f_{cpll} : f_{csb\_clk}$
CPMF	00x	Core PLL BYPASS / OFF
	010	1.0: 1
	011	1.5: 1
	100	2.0: 1
	101	2.5: 1
	110	3.0: 1
	111	3.5: 1

5.3 Clock Control Module

5.3.1 Memory Map/Register Definition

The clock configuration and status registers are shown in Table 5-5.

Table 5-5. Clock Configuration Registers memory map

Offset from CLOCK_BASE (0xFF40_0F00)[1]	Register	Access	Reset Value[2]	Section/Page
0x00	System PLL Mode Register (SPMR)	R	0x0U0U_0000	5.3.1.1/5-82
0x04	System Clock Control Register 1 (SCCR1)	R/W	0xE000_1C00	5.3.1.2/5-83
0x08	System Clock Control Register 2 (SCCR2)	R/W	0x2040_0000	5.3.1.3/5-85
0x0C	System Clock Frequency Register 1 (SCFR1)	R/W	0x0180_100C	5.3.1.4/5-86
0x10	System Clock Frequency Register 2 (SCFR2)	R/W	0xUU00_0808	5.3.1.6/5-89
0x14	System Clock Frequency Shadow Register 2 (SCFR2S)	R/W	0xUU0U_0000	5.3.1.6/5-89
0x18	Bread Crumb Register (BCR)	R/W	—[3]	5.3.1.7/5-90
0x1C	PSC0 Clock Control Register (P0CCR)	R/W	0xFFFE_0000	5.3.1.8/5-91
0x20	PSC1 Clock Control Register (P1CCR)	R/W	0xFFFE_0000	5.3.1.9/5-92
0x24	PSC2 Clock Control Register (P2CCR)	R/W	0xFFFE_0000	5.3.1.10/5-93
0x28	PSC3 Clock Control Register (P3CCR)	R/W	0xFFFE_0000	5.3.1.11/5-94
0x2C	PSC4 Clock Control Register (P4CCR)	R/W	0xFFFE_0000	5.3.1.12/5-95
0x30	PSC5 Clock Control Register (P5CCR)	R/W	0xFFFE_0000	5.3.1.13/5-95
0x34	PSC6 Clock Control Register (P6CCR)	R/W	0xFFFE_0000	5.3.1.14/5-96
0x38	PSC7 Clock Control Register (P7CCR)	R/W	0xFFFE_0000	5.3.1.15/5-97
0x3C	PSC8 Clock Control Register (P8CCR)	R/W	0xFFFE_0000	5.3.1.16/5-98
0x40	PSC9 Clock Control Register (P9CCR)	R/W	0xFFFE_0000	5.3.1.17/5-99
0x44–0x50	Reserved			
0x54	DIU Clock Config Register (DCCR)	R/W	0x0000_0000	5.3.1.18/5-100
0x58	MSCAN1 Clock Control Register (M1CCR)	R/W	0xFFFE_0000	5.3.1.19/5-101
0x5C	MSCAN2 Clock Control Register (M2CCR)	R/W	0xFFFE_0000	5.3.1.20/5-102
0x60	MSCAN3 Clock Control Register (M3CCR)	R/W	0xFFFE_0000	5.3.1.21/5-103
0x64	MSCAN4 Clock Control Register (M4CCR)	R/W	0xFFFE_0000	5.3.1.22/5-104
0x68–0x6F	Reserved			
0x70	OUT CLK0 Clock Configure Register (OUT0CCR)	R/W	0xFFFE_0000	5.3.1.23/5-104
0x74	OUT CLK1 Clock Configure Register (OUT1CCR)	R/W	0xFFFE_0000	5.3.1.24/5-105
0x78	OUT CLK2 Clock Configure Register (OUT2CCR)	R/W	0xFFFE_0000	5.3.1.25/5-106
0x7C	OUT CLK3 Clock Configure Register (OUT3CCR)	R/W	0xFFFE_0000	5.3.1.26/5-107
0x80	System Clock Frequency Register 3 (SCFR3)	R/W	0x2860_0000	5.3.1.27/5-108
0x84–0x8F	Reserved			

Table 5-5. Clock Configuration Registers memory map (continued)

Offset from CLOCK_BASE (0xFF40_0F00)[1]	Register	Access	Reset Value[2]	Section/Page
0x90	System PLL lock counter (SPLL_LOCK_CNT)	R	0x000U_UUUU	5.3.1.28/5-109
0x94–0xFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, "System Configuration and Memory Map (XLBMEN + Mem Map)."

2 In this column, the symbol "U" indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

3 Reset value is indeterminate.

5.3.1.1 System PLL Mode Register (SPMR)

Figure 5-6 shows the system PLL mode register. This is a read only register that retrieves its values from reset configuration word low loaded during the reset flow. This register is updated during a power up PORESET sequence.

Address: Base + 0x00 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SPMF				0	0	0	0	CPMF			
W																
Reset	0	0	0	0	—[1]	—[1]	—[1]	—[1]	0	0	0	0	—[1]	—[1]	—[1]	—[1]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-6. System PLL Mode Register (SPMR)

1 The reset value is defined by the latched reset configuration word (RST_CONF). See Table 4-2.

Table 5-6. SPMR field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>SPMF</td><td>System PLL Multiplication Factor. See Section 5.2.8.1, “System PLL Programming Model,” on page 5-79.
<table>
<tr><th>SPMF</th><th>$f_{spll} : f_{ref\_clk}$</th><th>SPMF</th><th>$f_{spll} : f_{ref\_clk}$</th></tr>
<tr><td>0000</td><td>68:1</td><td>1000</td><td>36:1</td></tr>
<tr><td>0001</td><td>PLL BYPASS[1]</td><td>1001</td><td>40:1</td></tr>
<tr><td>0010</td><td>12:1</td><td>1010</td><td>44:1</td></tr>
<tr><td>0011</td><td>16:1</td><td>1011</td><td>48:1</td></tr>
<tr><td>0100</td><td>20:1</td><td>1100</td><td>52:1</td></tr>
<tr><td>0101</td><td>24:1</td><td>1101</td><td>56:1</td></tr>
<tr><td>0110</td><td>28:1</td><td>1110</td><td>60:1</td></tr>
<tr><td>0111</td><td>32:1</td><td>1111</td><td>64:1</td></tr>
</table>
1 All the clocks from CLOCK block are bypassed with PLL reference clock.</td></tr>
<tr><td>CMPF</td><td>Core PLL Configuration. See Section 5.2.8.2, “e300 Core PLL Programming Model,” on page 5-80.
<table>
<tr><th>CPMF</th><th>$f_{cpll} : f_{csb\_clk}$</th><th>CPMF</th><th>$f_{cpll} : f_{csb\_clk}$</th></tr>
<tr><td>00x</td><td>36:1</td><td>100</td><td>2.0: 1</td></tr>
<tr><td>001</td><td>Core PLL BYPASS / OFF</td><td>101</td><td>2.5: 1</td></tr>
<tr><td>010</td><td>1.0: 1</td><td>110</td><td>3.0: 1</td></tr>
<tr><td>011</td><td>1.5: 1</td><td>111</td><td>3.5: 1</td></tr>
</table></td></tr>
</table>

5.3.1.2 System Clock Control Register 1 (SCCR1)

The System Clock Control Register 1 shown in Figure 5-7 controls device units with a configurable clock ratio.

Address: Base + 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CFG_EN	LPC_EN	NFC_EN	0	PSC0_EN	PSC1_EN	PSC2_EN	PSC3_EN	PSC4_EN	PSC5_EN	PSC6_EN	PSC7_EN	PSC8_EN	PSC9_EN	0	0
W																
Reset	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FIFOC_EN	0	FEC1_EN	0	0	DDR_EN	FEC2_EN	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	1[1]	1	1	0	0	0	0	0	0	0	0	0	0

Figure 5-7. System Clock Control Register 1 (SCCR1)

1 Bit is reserved, and must not be written or cleared. Value must remain 1.

Table 5-7. SCCR1 field descriptions

Field	Description
CFG_EN	This disables access to the memory map configuration registers for IO_CONTROL and MEMMAP configuration. 0 Disable 1 Enable
LPC_EN	lpc_clk Enable 0 Disable 1 Enable
NFC_EN	nfc_clk Enable 0 Disable 1 Enable
PSC0_EN	PSC0_clk Enable 0 Disable 1 Enable
PSC1_EN	PSC1 Clock Enable 0 Disable 1 Enable
PSC2_EN	PSC2 Clock Enable 0 Disable 1 Enable
PSC3_EN	PSC3 Clock Enable 0 Disable 1 Enable
PSC4_EN	PSC4 Clock Enable 0 Disable 1 Enable
PSC5_EN	PSC5 Clock Enable 0 Disable 1 Enable
PSC6_EN	PSC6 Clock Enable 0 Disable 1 Enable
PSC7_EN	PSC7 Clock Enable 0 Disable 1 Enable
PSC8_EN	PSC8 Clock Enable 0 Disable 1 Enable
PSC9_EN	PSC9 Clock Enable 0 Disable 1 Enable

Table 5-7. SCCR1 field descriptions (continued)

Field	Description
FIFOC_EN	FIFOC Clock Enable 0 Disable 1 Enable **Note:** If one of the PSC is used, the FIFOC clock must be enabled.
FEC1_EN	FEC1 Clock Enable 0 Disable 1 Enable
DDR_EN	MDDRC Clock Enable 0 Disable 1 Enable
FEC2_EN	FEC2 Clock Enable 0 Disable 1 Enable

5.3.1.3 System Clock Control Register 2 (SCCR2)

The System Clock Control Register 2 (SCCR2), shown in Figure 5-8, controls device units with a configurable clock ratio.

Address: Base + 0x08 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DIU _EN	0	MEM _EN	USB1 _EN	USB2 _EN	I2C _EN	AUTO _EN	SDHC1 _EN	0	0	0	0	IIM _ EN	0	SDHC 2_EN	0
W																
Reset	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-8. System Clock Control Register 2 (SCCR2)

Table 5-8. SCCR2 field descriptions

Field	Description
DIU_EN	DIU Clock Enable 0 Disable 1 Enable
MEM_EN	MEM Clock Enable 0 Disable 1 Enable
USB1_EN	USB1 Clock Enable 0 Disable 1 Enable
USB2_EN	USB2 Clock Enable 0 Disable 1 Enable
I2C_EN	I^2C Clock Enable 0 Disable 1 Enable
AUTO_EN	BDLC and MSCAN Clock Enable 0 Disable 1 Enable
SDHC1_EN	SDHC1 Clock Enable 0 Disable 1 Enable
IIM_EN	IIM Clock Enable 0 Disable 1 Enable
SDHC2_EN	SDHC2 Clock Enable 0 Disable 1 Enable

5.3.1.4 System Clock Frequency Register 1 (SCFR1)

The System Clock Frequency Register 1 (SCFR1), shown in Figure 5-9, controls device units with a configurable clock ratio.

Address: Base + 0x0C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	IPS_DIV			0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	LPC_DIV			0	0	0	DIU_DIV							
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0

Figure 5-9. System Clock Frequency Register 1 (SCFR1)

Table 5-9. SCFR1 field descriptions

Field	Description
IPS_DIV	IPS clock divide ratio. 000 Reserved. 001 Reserved. 010 Divide by 2. 011 Divide by 3 (default). 100 Divide by 4. 101 Reserved. 110 Divide by 6. 111 Reserved.
LPC_DIV	LocalPlus Bus Controller (LPC) clock divide ratio 000 Reserved. 001 Divide by 1. 010 Divide by 2 (default). 011 Divide by 3. 100 Divide by 4. 101 Reserved. 110 Reserved. 111 Reserved.

Table 5-9. SCFR1 field descriptions (continued)

Field	Description
DIU_DIV	Display Interface Unit (DIU) clock divide ratio

DIV	DIV value (decimal)	CSB_CLK:DIU_CLK
0b0000_1000	8	2[1]
0b0000_1100	12 (default)	3
0b0000_1101	13	3.25
0b0000_1110	14	3.5
0b0000_1111	15	3.75
0b0001_0000	16	4
0b0001_0001	17	4.25
. . .		
	n	*n*/4
. . .		
0b1111_1011	251	62.75
0b1111_1100	252	63
0b1111_1101	253	63.25
0b1111_1110	254	63.5
0b1111_1111	255	63.75

[1] This option should not be used.

5.3.1.5 System Clock Frequency Register 2 (SCFR2)

The System Clock Frequency Register 2 (SCFR2) shown in Figure 5-10 programs the SYS_DIV ratio. The values written to this register are visible after some update cycles. Changing this register value unlocks the Power Architecture PLL, because the source clock of the Power Architecture PLL is derived from the SYS_CLK.

Address: Base + 0x10 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYS_DIV						0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	—[1]	—[1]	—[1]	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SDHC2_DIV								SDHC1_DIV							
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

Figure 5-10. System Clock Frequency Register (SCFR2)

[1] The reset value is defined by the latched reset configuration word (RST_CONF). See Table 4-2.

Table 5-10. SCFR2 field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>SYS_DIV</td><td>SYS_CLK Divide Ratio
<table>
<tr><th>Divide Factor</th><th>Bit Encoding</th><th>Divide Factor</th><th>Bit Encoding</th><th>Divide Factor</th><th>Bit Encoding</th></tr>
<tr><td>2</td><td>00_0000</td><td>11</td><td>00_1011</td><td>23</td><td>01_0111</td></tr>
<tr><td>2.5</td><td>00_0001</td><td>12</td><td>00_1101</td><td>24</td><td>01_1001</td></tr>
<tr><td>3</td><td>00_0010</td><td>13</td><td>00_1110</td><td>25</td><td>01_1010</td></tr>
<tr><td>3.5</td><td>00_0011</td><td>14</td><td>01_0000</td><td>26</td><td>01_1100</td></tr>
<tr><td>4</td><td>00_0100</td><td>15</td><td>00_1111</td><td>27</td><td>01_1011</td></tr>
<tr><td>4.5</td><td>00_0101</td><td>16</td><td>01_0001</td><td>28</td><td>01_1101</td></tr>
<tr><td>5</td><td>00_0110</td><td>17</td><td>01_0010</td><td>29</td><td>01_1110</td></tr>
<tr><td>6</td><td>00_1000</td><td>18</td><td>01_0100</td><td>30</td><td>10_0000</td></tr>
<tr><td>7</td><td>00_0111</td><td>19</td><td>01_0011</td><td>31</td><td>01_1111</td></tr>
<tr><td>8</td><td>00_1001</td><td>20</td><td>01_0101</td><td>32</td><td>10_0001</td></tr>
<tr><td>9</td><td>00_1010</td><td>21</td><td>01_0110</td><td>33</td><td>10_0010</td></tr>
<tr><td>10</td><td>00_1100</td><td>22</td><td>01_1000</td><td colspan="2">All other settings are reserved.</td></tr>
</table></td></tr>
<tr><td>SDHC1_DIV</td><td>SDHC1 Divide Ratio
<table>
<tr><th>CSB_CLK:SDHC_CLK</th><th>Field value</th></tr>
<tr><td>2</td><td>0b0000_1000 (8d)</td></tr>
<tr><td>4</td><td>0b0001_0000 (16d)</td></tr>
<tr><td>6</td><td>0b0011_0000 (24d)</td></tr>
<tr><td>n/4</td><td>n</td></tr>
<tr><td>62</td><td>0b1111_1000 (248d)</td></tr>
</table>
Note: Only even integer divide ratios are supported.</td></tr>
<tr><td>SDHC2_DIV</td><td>SDHC2 Divide Ratio
<table>
<tr><th>CSB_CLK:SDHC_CLK</th><th>Field value</th></tr>
<tr><td>2</td><td>0b0000_1000 (8d)</td></tr>
<tr><td>4</td><td>0b0001_0000 (16d)</td></tr>
<tr><td>6</td><td>0b0011_0000 (24d)</td></tr>
<tr><td>n/4</td><td>n</td></tr>
<tr><td>62</td><td>0b1111_1000 (248d)</td></tr>
</table>
Note: Only even integer divide ratios are supported.</td></tr>
</table>

5.3.1.6 Shadow of System Clock Frequency Register 2 (SCFR2S)

The Shadow of System Clock Control Register 2 (SCFR2S), shown in Figure 5-11, programs the SYS_DIV ratio. When this register gets a control signal from the Power Management Control Module (PMC) to update the main SCFR2, the value of this shadow register overwrites the SCFR2.

Address: Base + 0x14 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYS_DIV						0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	—[1]	—[1]	—[1]	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-11. Shadow of System Clock Frequency Register (SCFR2S)

[1] The reset value is defined by the latched reset configuration word (RST_CONF). See Table 4-2.

Table 5-11. SCFR2S field descriptions

Field	Description
SYS_DIV	SYS_CLK Divide Ratio

Divide Factor	Bit Encoding	Divide Factor	Bit Encoding	Divide Factor	Bit Encoding
2	00_0000	11	00_1011	23	01_0111
2.5	00_0001	12	00_1101	24	01_1001
3	00_0010	13	00_1110	25	01_1010
3.5	00_0011	14	01_0000	26	01_1100
4	00_0100	15	00_1111	27	01_1011
4.5	00_0101	16	01_0001	28	01_1101
5	00_0110	17	01_0010	29	01_1110
6	00_1000	18	01_0100	30	10_0000
7	00_0111	19	01_0011	31	01_1111
8	00_1001	20	01_0101	32	10_0001
9	00_1010	21	01_0110	33	10_0010
10	00_1100	22	01_1000	All other settings are reserved.	

5.3.1.7 Bread Crumb Register (BCR)

The Bread Crumb Register (BCR), shown in Figure 5-12, provides a mechanism for retaining data after reset. Data in this register is not affected by $\overline{\text{PORESET}}$, $\overline{\text{HRESET}}$, or $\overline{\text{SRESET}}$.

Address: Base + 0x18 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-12. Bread Crumb Register (BCR)

Table 5-12. BCR field descriptions

Field	Description
BIT0x0	

5.3.1.8 PSC0 Clock Control Register (P0CCR)

The PSC0 clock control register, shown in Figure 5-13, controls the PSC0 MCLK divider ratio, the PSC0 MCLK divider enable, the PSC0 MCLK divider source, and the PSC0 MCLK source. Table 5-13 defines the bit fields of P0CCR.

Address: Base + 0x1C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PSC0_MCLK_DIV															MCLK_0_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PSC0_MCLK_0_SRC		0	0	0	0	0	0	PSC0_MCLK_1_SRC	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-13. PSC0 Clock Control Register (P0CCR)

Table 5-13. P0CCR field descriptions

Field	Description
PSC0_ MCLK_DIV	MCLK_DIV Divider Ratio $f_{mclk\_out} = f_{mclk\_src} / (MCLK\_DIV + 1)$ **Note:** The maximum supported frequency for f_{mclk_out} is the IP Bus frequency. A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of MCLK_EN = 0.

Table 5-13. P0CCR field descriptions (continued)

Field	Description
MCLK0_EN	PSC0 Divider Enable 0 PSC0 divider is disabled. 1 PSC0 divider is enabled.
PSC0_MCLK_0_SRC	PSC MCLK Divider Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.
PSC0_MCLK_1_SRC	PSC MCLK Source 0 MCLK_DIV. 1 Reserved.

5.3.1.9 PSC1 Clock Control Register (P1CCR)

The PSC1 clock control register, shown in Figure 5-14, controls the PSC1 MCLK divider ratio, the PSC1 MCLK divider enable, the PSC1 MCLK divider source, and the PSC1 MCLK source. Table 5-14 defines the bit fields of P1CCR.

Address: Base + 0x20 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PSC1_MCLK_DIV															MCLK1_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSC1_MCLK_0_SRC		0	0	0	0	0	0	PSC1_MCLK_1_SRC	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-14. PSC1 Clock Control Register (P1CCR)

Table 5-14. P1CCR field descriptions

Field	Description
PSC1_MCLK_DIV	MCLK_DIV Divider Ratio $f_{mclk\_out} = f_{mclk\_src} / (MCLK\_DIV + 1)$ **Note:** The maximum supported frequency for f_{mclk_out} is the IP Bus frequency. A value of 0x000 bypasses the divider. **Note:** This value can only be changed when the value of MCLK_EN equals 0.
MCLK1_EN	PSC1 Divider Enable 0 PSC1 divider is disabled. 1 PSC1 divider is enabled.

Table 5-14. P1CCR field descriptions (continued)

Field	Description
PSC1_ MCLK_0_SRC	PSC MCLK Divider Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.
PSC1_ MCLK_1_SRC	PSC MCLK Source 0 MCLK_DIV. 1 Reserved.

5.3.1.10 PSC2 Clock Control Register (P2CCR)

The PSC2 Clock Control Register (P2CCR), shown in Figure 5-15, controls the PSC2 MCLK divider ratio, the PSC2 MCLK divider enable, the PSC2 MCLK divider source, and the PSC2 MCLK source. Table 5-15 defines the bit fields of P2CCR.

Address: Base + 0x24 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PSC2_MCLK_DIV															MCLK 2_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSC2_MCLK _0_SRC		0	0	0	0	0	0	PSC2_ MCLK_ 1_SRC	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-15. PSC2 Clock Control Register (P2CCR)

Table 5-15. P2CCR field descriptions

Field	Description
PSC2_ MCLK_DIV	MCLK_DIV Divider Ratio $f_{mclk\_out} = f_{mclk\_src} / (MCLK\_DIV + 1)$ **Note:** The maximum supported frequency for f_{mclk_out} is the IP Bus frequency. A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of MCLK_EN equals 0.
MCLK2_EN	PSC2 Divider Enable 0 PSC2 divider is disabled. 1 PSC2 divider is enabled.
PSC2_ MCLK_0_SRC	PSC MCLK Divider Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.

Table 5-15. P2CCR field descriptions (continued)

Field	Description
PSC2_ MCLK_1_SRC	PSC MCLK Source 0 MCLK_DIV. 1 Reserved.

5.3.1.11 PSC3 Clock Control Register (P3CCR)

The PSC3 Clock Control Register (P3CCR), shown in Figure 5-16, controls the PSC3 MCLK divider ratio, the PSC3 MCLK divider enable, the PSC3 MCLK divider source, and the PSC3 MCLK source. Table 5-16 defines the bit fields of P3CCR.

Address: Base + 0x28 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PSC3_MCLK_DIV															MCLK 3_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSC3_MCLK _0_SRC		0	0	0	0	0	0	PSC3_ MCLK_ 1_SRC	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-16. PSC3 Clock Control Register (P3CCR)

Table 5-16. P3CCR field descriptions

Field	Description
PSC3_ MCLK_DIV	MCLK_DIV Divider Ratio $f_{mclk\_out} = f_{mclk\_src} / (MCLK\_DIV + 1)$ **Note:** The maximum supported frequency for f_{mclk_out} is the IP Bus frequency. A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of MCLK_EN equals 0.
MCLK3_EN	PSC3 Divider Enable 0 PSC3 divider is disabled. 1 PSC3 divider is enabled.
PSC3_ MCLK_0_SRC	PSC MCLK Divider Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.
PSC3_ MCLK_1_SRC	PSC MCLK Source 0 MCLK_DIV. 1 Reserved.

5.3.1.12 PSC4 Clock Control Register (P4CCR)

The PSC4 Clock Control Register (P4CCR), shown in Figure 5-17, controls the PSC4 MCLK divider ratio, the PSC4 MCLK divider enable, the PSC4 MCLK divider source, and the PSC4 MCLK source. Table 5-17 defines the bit fields of P4CCR.

Address: Base + 0x2C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PSC4_MCLK_DIV															MCLK 4_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSC4_MCLK _0_SRC		0	0	0	0	0	0	PSC4_ MCLK_ 1_SRC	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-17. PSC4 Clock Control Register (P4CCR)

Table 5-17. P4CCR field descriptions

Field	Description
PSC4_ MCLK_DIV	MCLK_DIV Divider Ratio $f_{mclk\_out} = f_{mclk\_src} / (MCLK\_DIV + 1)$ **Note:** The maximum supported frequency for f_{mclk_out} is the IP Bus frequency. A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of MCLK_EN equals 0.
MCLK4_EN	PSC4 Divider Enable 0 PSC4 divider is disabled. 1 PSC4 divider is enabled.
PSC4_ MCLK_0_SRC	PSC MCLK Divider Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.
PSC4_ MCLK_1_SRC	PSC MCLK Source 0 MCLK_DIV. 1 Reserved.

5.3.1.13 PSC5 Clock Control Register (P5CCR)

The PSC5 Clock Control Register (P5CCR), shown in Figure 5-18, controls the PSC5 MCLK divider ratio, the PSC5 MCLK divider enable, the PSC5 MCLK divider source, and the PSC5 MCLK source. Table 5-18 defines the bit fields of P5CCR.

Address: Base + 0x30 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PSC5_MCLK_DIV															MCLK 5_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSC5_MCLK _0_SRC		0	0	0	0	0	0	PSC5_ MCLK_ 1_SRC	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-18. PSC5 Clock Control Register (P5CCR)

Table 5-18. P5CCR field descriptions

Field	Description
PSC5_ MCLK_DIV	MCLK_DIV Divider Ratio $f_{mclk\_out} = f_{mclk\_src} / (MCLK\_DIV + 1)$ **Note:** The maximum supported frequency for f_{mclk_out} is the IP Bus frequency. A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of MCLK_EN equals 0.
MCLK5_EN	PSC5 Divider Enable 0 PSC52 divider is disabled. 1 PSC5 divider is enabled.
PSC5_ MCLK_0_SRC	PSC MCLK Divider Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.
PSC5_ MCLK_1_SRC	PSC MCLK Source 0 MCLK_DIV. 1 Reserved.

5.3.1.14 PSC6 Clock Control Register (P6CCR)

The PSC6 Clock Control Register (P6CCR), shown in Figure 5-19, controls the PSC6 MCLK divider ratio, the PSC6 MCLK divider enable, the PSC6 MCLK divider source, and the PSC6 MCLK source. Table 5-19 defines the bit fields of P6CCR.

Address: Base + 0x34 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PSC6_MCLK_DIV															MCLK 6_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PSC6_MCLK _0_SRC								PSC6_ MCLK_ 1_SRC	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-19. PSC6 Clock Control Register (P6CCR)

Table 5-19. P6CCR field descriptions

Field	Description
PSC6_ MCLK_DIV	MCLK_DIV Divider Ratio $f_{mclk\_out} = f_{mclk\_src} / (MCLK\_DIV + 1)$ **Note:** The maximum supported frequency for f_{mclk_out} is the IP Bus frequency. A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of MCLK_EN equals 0.
MCLK6_EN	PSC6 Divider Enable 0 PSC6 divider is disabled. 1 PSC6 divider is enabled.
PSC6_ MCLK_0_SRC	PSC MCLK Divider Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.
PSC6_ MCLK_1_SRC	PSC MCLK Source 0 MCLK_DIV. 1 Reserved.

5.3.1.15 PSC7 Clock Control Register (P7CCR)

The PSC7 Clock Control Register (P7CCR), shown in Figure 5-20, controls the PSC7 MCLK divider ratio, the PSC7 MCLK divider enable, the PSC7 MCLK divider source, and the PSC7 MCLK source. Table 5-20 defines the bit fields of P7CCR.

Address: Base + 0x38 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PSC7_MCLK_DIV															MCLK 7_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSC7_MCLK _0_SRC		0	0	0	0	0	0	PSC7_ MCLK_ 1_SRC	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-20. PSC7 Clock Control Register (P7CCR)

Table 5-20. P7CCR field descriptions

Field	Description
PSC7_ MCLK_DIV	MCLK_DIV Divider Ratio $f_{mclk\_out} = f_{mclk\_src} / (MCLK\_DIV + 1)$ **Note:** The maximum supported frequency for f_{mclk_out} is the IP Bus frequency. A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of MCLK_EN equals 0.
MCLK7_EN	PSC7 Divider Enable 0 PSC7 divider is disabled. 1 PSC7 divider is enabled.
PSC7_ MCLK_0_SRC	PSC MCLK Divider Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.
PSC7_ MCLK_1_SRC	PSC MCLK Source 0 MCLK_DIV. 1 Reserved.

5.3.1.16 PSC8 Clock Control Register (P8CCR)

The PSC8 Clock Control Register (P8CCR), shown in Figure 5-21, controls the PSC8 MCLK divider ratio, PSC8 MCLK divider enable, PSC8 MCLK divider source, and the PSC8 MCLK source. Table 5-21 defines the bit fields of P8CCR.

Address: Base + 0x3C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PSC8_MCLK_DIV															MCLK 8_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSC8_MCLK _0_SRC		0	0	0	0	0	0	PSC8_ MCLK_ 1_SRC	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-21. PSC8 Clock Control Register (P8CCR)

Table 5-21. P8CCR field descriptions

Field	Description
PSC8_ MCLK_DIV	MCLK_DIV Divider Ratio $f_{mclk\_out} = f_{mclk\_src} / (MCLK\_DIV + 1)$ **Note:** The maximum supported frequency for f_{mclk_out} is the IP Bus frequency. A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of MCLK_EN equals 0.
MCLK8_EN	PSC8 Divider Enable 0 PSC8 divider is disabled. 1 PSC8 divider is enabled.
PSC8_ MCLK_0_SRC	PSC MCLK Divider Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.
PSC8_ MCLK_1_SRC	PSC MCLK Source 0 MCLK_DIV. 1 Reserved.

5.3.1.17 PSC9 Clock Control Register (P9CCR)

The PSC9 Clock Control Register (P9CCR), shown in Figure 5-22, controls the PSC9 MCLK divider ratio, PSC9 MCLK divider enable, PSC9 MCLK divider source, and the PSC9 MCLK source.

Table 5-22 defines the bit fields of P9CCR.

Address: Base + 0x40 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PSC9_MCLK_DIV															MCLK 9_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSC9_MCLK _0_SRC		0	0	0	0	0	0	PSC9_ MCLK_ 1_SRC	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-22. PSC9 Clock Control Register 9 (P9CCR)

Table 5-22. P9CCR field descriptions

Field	Description
PSC9_ MCLK_DIV	MCLK_DIV Divider Ratio $f_{mclk\_out} = f_{mclk\_src} / (MCLK\_DIV + 1)$ **Note:** The maximum supported frequency for f_{mclk_out} is the IP Bus frequency. A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of MCLK_EN equals 0.
MCLK9_EN	PSC9 Divider Enable 0 PSC9 divider is disabled. 1 PSC9 divider is enabled.
PSC9_ MCLK_0_SRC	PSC MCLK Divider Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.
PSC9_ MCLK_1_SRC	PSC MCLK Source 0 MCLK_DIV. 1 Reserved.

5.3.1.18 DIU Clock Config Register (DCCR)

The DIU Clock Config Register (DCCR), shown in Figure 5-23, configures the number of Coherent Systems Bus (CSB) cycles delay added to the pixel clock to pad compare to pixel clock to DIU block, and whether the DIU pixel clock is inverted. Table 5-23 defines the bit fields of DCCR.

Address: Base + 0x54 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DLY_NUM							CLK_INV	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-23. DIU Clock Config Register (DCCR)

Table 5-23. DCCR field descriptions

Field	Description
DLY_NUM	Number of CSB_CLK cycles delay added to pixel clock output to pad compare pixel clock to DIU. 00 0 cycle delay. 01 2 cycles delay. 10 4 cycles delay. 11 6 cycles delay.
CLK_INV	Pixel Clock Inversion 0 The pixel clock to pad is the same as the one to the DIU. The DLY_NUM bitfield specifies s the delay CSB cycles added to it. 1 The pixel clock to pad is inverted from the one to the DIU. The DLY_NUM bitfield specifies s the delay CSB cycles added to it.

5.3.1.19 MSCAN1 Clock Control Register (M1CCR)

The MSCAN1 clock control register shown in Figure 5-24 controls the MSCAN1 source clock divider ratio, the MSCAN1 divider enable, and the MSCAN1 divider clock source. Table 5-24 defines the bit fields of M1CCR.

Address: Base + 0x58 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSCAN1_CLK_DIV															MSCAN1_EN
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MSCAN1_CLK_SRC		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-24. MSCAN1 Clock Control Register (M1CCR)

Table 5-24. M1CCR field descriptions

Field	Description
MSCAN1_CLK_DIV	MSCAN1 divider Ratio $f_{can\_source\_clk} = f_{mscan\_src} / (MSCAN1\_CLK\_DIV + 1)$ A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of MSCAN1_EN equals 0
MSCAN1_EN	MSCAN1 Divider Enable 0 MSCAN1 divider is disabled. 1 MSCAN1 divider is enabled.
MSCAN1_CLK_SRC	MSCAN1 CLK Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.

5.3.1.20 MSCAN2 Clock Control Register (M2CCR)

The MSCAN2 clock control register shown in Figure 5-25 controls the MSCAN2 source clock divider ratio, MSCAN2 divider enable, and the MSCAN2 divider clock source. Table 5-25 defines the bit fields of M2CCR.

Address: Base + 0x5C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MSCAN2_CLK_DIV															MSCAN2_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MSCAN2_CLK_SRC		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-25. MSCAN2 Clock Control Register (M2CCR)

Table 5-25. M2CCR field descriptions

Field	Description
MSCAN2_CLK_DIV	MSCAN2 divider Ratio $f_{can\_source\_clk} = f_{mscan\_src} / (MSCAN2\_CLK\_DIV + 1)$ A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of MSCAN2_EN equals 0.
MSCAN2_EN	MSCAN2 Divider Enable 0 MSCAN2 divider is disabled. 1 MSCAN2 divider is enabled.

Table 5-25. M2CCR field descriptions (continued)

Field	Description
MSCAN2_ CLK_SRC	MSCAN2 CLK Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.

5.3.1.21 MSCAN3 Clock Control Register (M3CCR)

The MSCAN3 clock control register shown in Figure 5-26 controls the MSCAN3 source clock divider ratio, the MSCAN3 divider enable, and the MSCAN3 divider clock source. Table 5-26 defines the bit fields of M3CCR.

Address: Base + 0x60 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MSCAN3_CLK_DIV															MSCAN3_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MSCAN3_CLK_SRC		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-26. MSCAN3 Clock Control Register (M3CCR)

Table 5-26. M3CCR field descriptions

Field	Description
MSCAN3_CLK_ DIV	MSCAN3 divider Ratio $f_{can\_source\_clk} = f_{mscan\_src} / (MSCAN3\_CLK\_DIV + 1)$ A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of MSCAN3_EN equals 0.
MSCAN3_EN	MSCAN3 Divider Enable 0 MSCAN3 divider is disabled. 1 MSCAN3 divider is enabled.
MSCAN3_ CLK_SRC	MSCAN3 CLK Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.

5.3.1.22 MSCAN4 Clock Control Register (M4CCR)

The MSCAN4 clock control register shown in Figure 5-27 controls the MSCAN4 source clock divider ratio, the MSCAN4 divider enable, and the MSCAN4 divider clock source. Table 5-27 defines the bit fields of M4CCR.

Address: Base + 0x64 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MSCAN4_CLK_DIV															MSCAN4_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MSCAN4_CLK_SRC		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-27. MSCAN4 Clock Control Register (M4CCR)

Table 5-27. M4CCR field descriptions

Field	Description
MSCAN4_CLK_DIV	MSCAN4 divider Ratio $f_{can\_source\_clk} = f_{mscan\_src} / (MSCAN4\_CLK\_DIV + 1)$ A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of MSCAN4_EN equals 0.
MSCAN4_EN	MSCAN4 Divider Enable 0 MSCAN4 divider is disabled. 1 MSCAN4 divider is enabled.
MSCAN4_CLK_SRC	MSCAN4 CLK Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.

5.3.1.23 Out Clock 0 Config Register (OUT0CCR)

The Out Clock 0 clock config register is shown in Figure 5-28. Table 5-28 defines the bit fields of OUT0CCR.

Address: Base + 0x70 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OUT0_DIV															OUT0_EN
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUT0_CLK_SRC		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-28. Out Clock 0 Clock Config Register (OUT0CCR)

Table 5-28. OUT0CCR field descriptions

Field	Description
OUT0_DIV	OUT0 divider Ratio $f_{out0\_clk} = f_{out0\_clk\_src} / (OUT0\_DIV + 1)$ A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of OUT0_EN equals 0
OUT0_EN	OUT0 Divider Enable 0 OUT0 divider is disabled. 1 OUT0 divider is enabled.
OUT0_CLK_SRC	Out Clock 0 Clock Source 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.

5.3.1.24 Out Clock 1 Config Register (OUT1CCR)

The Out Clock 1 clock config register is shown in Figure 5-29. Table 5-29 defines the bit fields of OUT1CCR.

Address: Base + 0x74 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OUT1_DIV															OU1_EN
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUT1_CLK_SRC		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-29. Out Clock 1 Clock Config Register (OUT1CCR)

Table 5-29. OUT1CC field descriptions

Field	Description
OUT1_DIV	OUT1 divider Ratio $f_{out1\_clk} = f_{out1\_clk\_src} / (OUT1\_DIV + 1)$ A value of 0x0 will bypass the divider. **Note:** This value can only be changed when the value of OUT1_EN equals 0.
OUT1_EN	OUT1 Divider Enable 0 OUT1 divider is disabled. 1 OUT1 divider is enabled.
OUT1_CLK_SRC	Out Clock 1 Clock Source. 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.

5.3.1.25 Out Clock 2 Config Register (OUT2CCR)

The Out Clock 2 clock config register is shown in Figure 5-30. Table 5-30 defines the bit fields of OUT2CCR.

Address: Base + 0x78 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	OUT2_DIV															OUT2_EN
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUT2_CLK_SRC		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-30. Out Clock 2 Clock Config Register (OUT2CCR)

Table 5-30. OUT2CCR field descriptions

Field	Description
OUT2_DIV	OUT2 divider Ratio $f_{out2\_clk} = f_{out2\_clk\_src} / (OUT2\_DIV + 1)$ A value of 0x0 will bypass the divider **Note:** This value can only be changed when the value of OUT2_EN equals 0
OUT2_EN	OUT2 Divider Enable 0 OUT2 divider is disabled 1 OUT2 divider is enabled

Table 5-30. OUT2CCR field descriptions (continued)

Field	Description
OUT2_CLK_SRC	Out Clock 2Clock Source. 00 From SYS_CLK 01 From REF_CLK 10 From PSC_MCLK_IN 11 From CAN_CLK_IN

5.3.1.26 Out Clock 3 Config Register (OUT3CCR)

The Out Clock 3 clock config register is shown in Figure 5-31. Table 5-31 defines the bit fields of OUT3CCR.

Address: Base + 0x7C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OUT3_DIV															OUT3_EN
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUT3_CLK_SRC		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-31. Out Clock 3 Clock Config Register (OUT3CCR)

Table 5-31. OUT3CCR field descriptions

Field	Description
OUT3_DIV	OUT3 divider Ratio $f_{out3\_clk} = f_{out3\_clk\_src} / (OUT3\_DIV + 1)$ A value of 0x0000 bypasses the divider. **Note:** This value can only be changed when the value of OUT3_EN equals 0.
OUT3_EN	OUT3 Divider Enable 0 OUT3 divider is disabled. 1 OUT3 divider is enabled.
OUT3_CLK_SRC	Out Clock 3 Clock Source. 00 From SYS_CLK. 01 From REF_CLK. 10 From PSC_MCLK_IN. 11 From CAN_CLK_IN.

5.3.1.27 System Clock Frequency Register (SCFR3)

The System Clock Frequency Register (SCFR3) is shown in Figure 5-32. The NFC fractional clock divider controlled by this register provides a clock with a programmable high and low phase.

Address: Base + 0x80 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NFC_RATIO_H								NFC_RATIO_L							
Reset	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-32. System Clock Frequency Register (SCFR3)

Table 5-32. SCFR3 field descriptions

Field	Description
NFC_RATIO_H	NFC fractional clock divider ratio. The clock high phase is defined by the NFC_RATIO_H parameter, the low phase by NFC_RATIO_L. The divide values for the high phase of the clock are as follows: **CSB_CLK:NFC_CLK** / **Field value** 3 / 0b0000_1100 (12d) 3.25 / 0b0000_1101 (13d) 3.5 / 0b0000_1110 (14d) 3.75 / 0b0000_1111 (15d) 4 / 0b0001_0000 (16d) 4.25 / 0b0001_0001 (17d) *n*/4 / *n* 10 (default) / 0b0010_1000 (40d) *n*/4 / *n* 62.75 / 0b1111_1011 (251d) 63 / 0b1111_1100 (252d) 63.25 / 0b1111_1101 (253d) 63.5 / 0b1111_1110 (254d) 63.75 / 0b1111_1111 (255d) The NFC flash_clock frequency = SYS_CLK/ (NFC_RATIO_H/4 + NFC_RATIO_L/4), with the reset value: NFC flash_clock is 400/(40/4 + 96/4) = 11.76 MHz.

Table 5-32. SCFR3 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>NFC_RATIO_L</td><td>NFC fractional clock divider ratio. The clock high phase is defined by the NFC_RATIO_H parameter, the low phase by NFC_RATIO_L. The divide values for the low phase of the clock are as follows:
<table>
<tr><th>CSB_CLK:NFC_CLK</th><th>Field value</th></tr>
<tr><td>3</td><td>0b0000_1100 (12d)</td></tr>
<tr><td>3.25</td><td>0b0000_1101 (13d)</td></tr>
<tr><td>3.5</td><td>0b0000_1110 (14d)</td></tr>
<tr><td>3.75</td><td>0b0000_1111 (15d)</td></tr>
<tr><td>4</td><td>0b0001_0000 (16d)</td></tr>
<tr><td>4.25</td><td>0b0001_0001 (17d)</td></tr>
<tr><td>n/4</td><td>n</td></tr>
<tr><td>24 (default)</td><td>0b0110_0000 (96d)</td></tr>
<tr><td>n/4</td><td>n</td></tr>
<tr><td>62.75</td><td>0b1111_1011 (251d)</td></tr>
<tr><td>63</td><td>0b1111_1100 (252d)</td></tr>
<tr><td>63.25</td><td>0b1111_1101 (253d)</td></tr>
<tr><td>63.5</td><td>0b1111_1110 (254d)</td></tr>
<tr><td>63.75</td><td>0b1111_1111 (255d)</td></tr>
</table>
The NFC flash_clock frequency = SYS_CLK/ (NFC_RATIO_H/4 + NFC_RATIO_L/4), with the reset value: NFC flash_clock is 400/(40/4 + 96/4) = 11.76 MHz.</td></tr>
</table>

5.3.1.28 System PLL Lock Counter Register (SPLL_LOCK_CNT)

The system PLL lock counter register 3 is shown in Figure 5-33.

Address: Base + 0x90 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	SYS PLL_ LOCK _STAT US	SYS PLL _UN LOCK	SYS PLL_ LOCK
W																
Reset[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	U	U	U

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SYSPLL_LOCK_CNT															
W																
Reset	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Figure 5-33. System PLL Lock Counter Register (SPLL_LOCK_CNT)

[1] Reset value depends on the lock time and status of the system, and is indeterminate.

Table 5-33. SPLL_LOCK_CNT field descriptions

Field	Description
SYSPLL_LOCK_STATUS	The SYSPLL_LOCK_STATUS shows the current status of the system PLL lock detector. 0 System PLL is unlocked. 1 System PLL is locked.
SYSPLL_UNLOCK	The SYSPLL_UNLOCK captures the first unlock event after a lock event. This bit is cleared by $\overline{\text{HRESET}}$. 0 No unlock event has occurred after the last lock event. 1 An unlock event has occurred after the last lock event.
SYSPLL_LOCK	The SYSPLL_LOCK captures the first lock event after $\overline{\text{HRESET}}$. This bit is cleared by $\overline{\text{HRESET}}$. 0 No lock event occurred. 1 A lock event occurred.
SYSPLL_LOCK_CNT	The register contains the number of REF_CLK cycles between the $\overline{\text{PORESET}}$ deassertion and the System PLL lock detector indicating the first time a lock state. (SYSPLL_LOCK = 1). This bitfield is cleared by $\overline{\text{HRESET}}$. **Note:** The reset value of this bitfield is indeterminate.

Chapter 6
Byte Data Link Controller (BDLC)

6.1 Introduction

The Byte Data Link Controller (BDLC) is a serial communication module that allows the user to send and receive messages across a Society of Automotive Engineers (SAE) J1850 serial communication network. The user's software manages each transmitted or received message on a byte-by-byte basis, while the BDLC performs all of the network access, arbitration, message framing and error detection duties.

It is recommended that the reader be familiar with the operation and requirements of the SAE J1850 protocol as described in SAE Standard J1850 Class B Data Communications Network Interface document prior to proceeding with this specification.

The BDLC module is designed in a modular structure for use as an IP block. A general working knowledge of the IP bus signals and bus control is assumed in the writing of this document.

Figure 6-1 shows the organization of the BDLC module. The Tx/Rx shadow register function as Buffers provide storage for data received and data to be transmitted onto the J1850 bus. The Protocol Handler is responsible for the encoding and decoding of data bits and special message symbols during transmission and reception. The MUX Interface provides the link between the BDLC digital section and the analog Physical Interface. The wave shaping, driving and digitizing of data is performed by the Physical Interface.

The Physical Interface is not implemented in the BDLC module and must be provided externally.

The main functional blocks of the BDLC module are explained in greater detail in the following sections.

Use of the BDLC module in message networking fully implements the SAE Standard J1850 Class B Data Communication Network Interface specification.

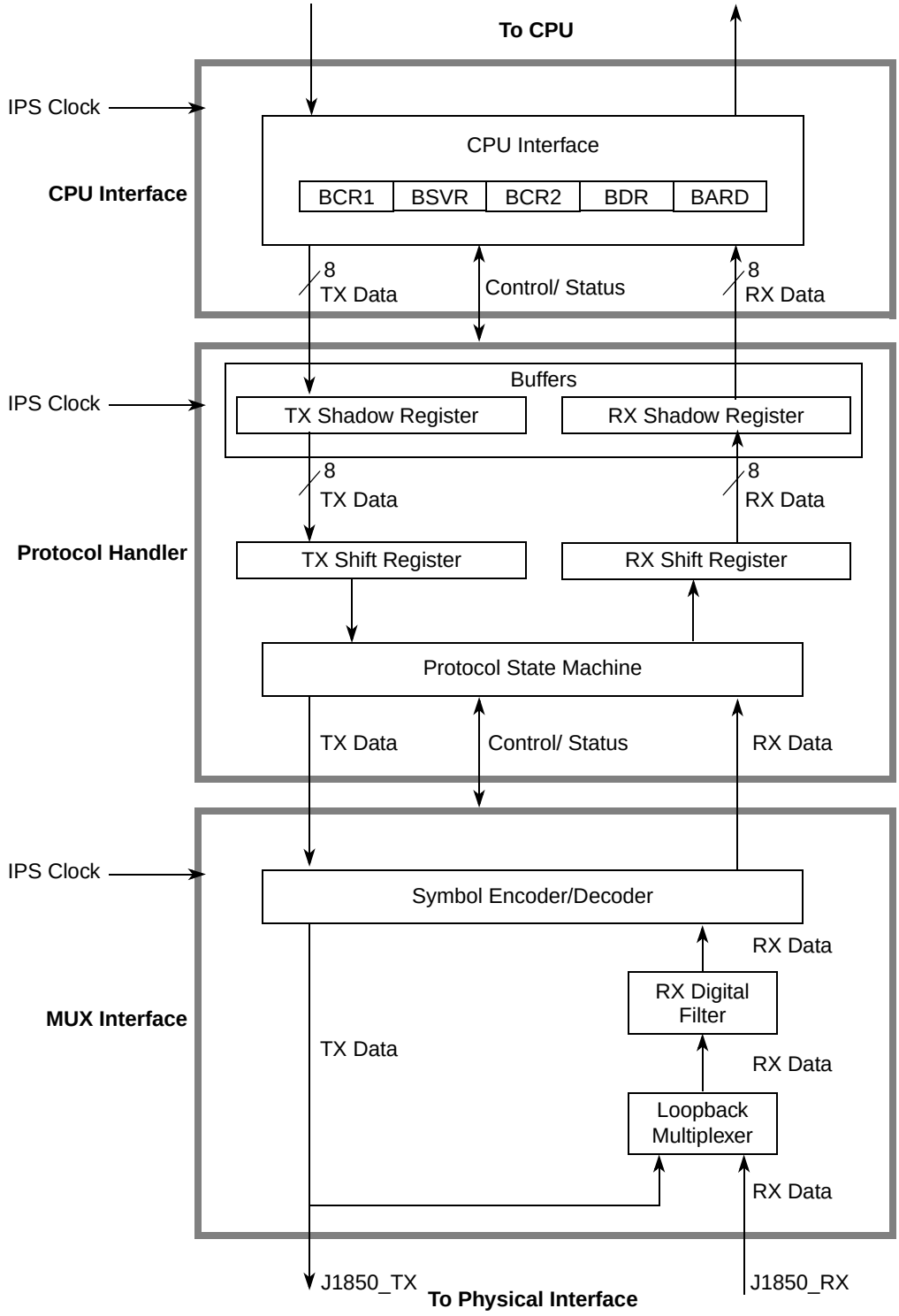


Figure 6-1. BDLC Block Diagram

6.1.1 Features

Features of the BDLC module include the following:

- SAE J1850 Class B data communications network interface compatible and ISO compatible for low-speed (≤ 125 kbit/s) serial data communications in automotive applications
- 10.4 kbit/s Variable Pulse Width (VPW) bit format
- Digital noise filter
- Digital loopback mode
- 4× receive and transmit mode, 41.6 kbit/s supported
- BREAK symbol generation Supported
- Block mode receive and transmit supported
- Collision detection
- Hardware Cyclical Redundancy Check (CRC) generation and checking
- Dedicated register for symbol timing adjustments
- In-Frame Response (IFR) types 0, 1, 2, and 3 supported
- Polling and CPU interrupt generation with vector lookup available

6.2 External Signal Description

The BDLC module has two external pins.

Table 6-1. Signal Properties

Name	Port	Function	I/O	Reset	Pull Up
J1850_TX	Output	The J1850_TX pin serves as the transmit output channel for the BDLC module	O	0	—
J1850_RX	Input	The J1850_RX pin serves as the receive input channel for the BDLC module	I		—

6.3 Memory Map and Register Definition

6.3.1 Memory Map

The BDLC memory map is shown in Table 6-2.

Table 6-2. BDLC memory map

Offset from BDLC_BASE (0xFF40_1400)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x00	BDLC Control Register 1 (BDLC_DLCBCR1)	R/W	0xC0	6.3.2.1/6-115
0x01	BDLC State Vector Register (BDLC_DLCBSVR)	R	0x00	6.3.2.2/6-116
0x02–0x03	Reserved			
0x04	BDLC Control Register 2 (BDLC_DLCBCR2)	R/W	0x40	6.3.2.3/6-118
0x05	BDLC Data Register (BDLC_DLCBDR)	R/W	0x00	6.3.2.4/6-123

Table 6-2. BDLC memory map (Continued)

Offset from BDLC_BASE (0xFF40_1400)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x06–0x07	Reserved			
0x08	BDLC Analog Round Trip Delay Register (BDLC_DLCBARD)	R/W	0x50	6.3.2.5/6-124
0x09	BDLC Rate Select Register (BDLC_DLCBRSR)	R/W	0x00	6.3.2.6/6-126
0x0A–0x0B	Reserved			
0x0C	BDLC Control Register (BDLC_DLCSCR)	R/W	0x00	6.3.2.7/6-127
0x0D	BDLC Status Register (BDLC_DLCBSTAT)	R/W	0x00	6.3.2.8/6-128
0x0E–0xFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, "System Configuration and Memory Map (XLBMEN + Mem Map)."

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

3 In this column, the symbol "U" indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

6.3.2 Register Summary

Table 6-3. BDLC Register Summary

Name		7	6	5	4	3	2	1	0
0x00 BDLC_ DLCBCR1	R	IMSG	CLKS	0	0	0	0	IE	0
	W								
0x01 BDLC_ DLCBSVR	R	0	0	I3	I2	I1	I0	0	0
	W								
0x04 BDLC_ DLCBCR2	R	SMRST	DLOOP	4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
	W								
0x05 BDLC_ DLCBDR	R	D7	D6	D5	D4	D3	D2	D1	D0
	W								
0x08 BDLC_ DLCBARD	R	0	RXPOL	0	BO4	BO3	BO2	BO1	BO0
	W								
0x09 BDLC_ DLCBRSR	R	R7	R	R5	R4	R3	R2	R1	R0
	W								
0x0C BDLC_ DLCSCR	R	0	0	0	BDLCE	0	0	0	BREAK
	W								

Table 6-3. BDLC Register Summary (Continued)

Name		7	6	5	4	3	2	1	0
0x0D BDLC_ DLCBSTAT	R	0	0	0	0	0	tst_divte_t4	tst_crcv_t4	IDLE
	W								

6.3.2.1 BDLC Control Register 1 (BDLC_DLCBCR1)

The BDLC Control Register 1 (BDLC_DLCBCR1) is used to configure and control the BDLC module.

Address: Base + 0x00 Access: User read/write

	0	1	2	3	4	5	6	7
R	IMSG	CLKS	0	0	0	0	IE	0
W								
Reset	1	1	0	0	0	0	0	0

Figure 6-2. BDLC Control Register 1 (BDLC_DLCBCR1)

Table 6-4. BDLC_DLCBCR1 field descriptions

Field	Description
IMSG	Ignore Message This bit allows the CPU to ignore messages by disabling updates of the BDLC State Vector Register (BDLC_DLCBSVR) until a new Start of Frame (SOF) or a BREAK symbol is detected. BDLC module transmitter and receiver operation are unaffected by the state of the IMSG bit. There are three situations in which interrupts are not masked by the IMSG bit: when a wakeup interrupt occurs, when TX data register empty interrupt occurs, and when a receiver error occurs, which causes a byte pending transmission to be flushed from the transmit shadow register. See Section 6.3.2.4, "BDLC Data Register (BDLC_DLCBDR)," for a description of the conditions that cause a pending transmission to be flushed. 0 Enable BDLC State Vector Register Updates. This bit is automatically cleared by the reception of a SOF symbol or a BREAK symbol. It then allows updates of the state vector register to occur. 1 Disable BDLC State Vector Register Updates. When set, all BDLC interrupt sources (besides the exceptions previously described) are prevented from updating the BDLC State Vector Register status bits. This e behavior is described in Section 6.3.2.2, "BDLC State Vector Register (BDLC_DLCBSVR)." Setting IMSG does not clear pending interrupt flags. If this bit is set while the BDLC is receiving or transmitting a message, state vector register updates are inhibited for the rest of the message.
CLKS	Clock Select The nominal BDLC operating frequency (MUX interface clock frequency - f_{bdlc}) must always be 1.048576 MHz or 1 MHz for J1850 bus communications to take place properly. The CLKS register bit is provided to allow the user to indicate to the BDLC module which frequency (1.048576 MHz or 1 MHz) is used so that each symbol time can be automatically adjusted. The CLKS bit is a write-once bit. All writes to this bit are ignored after the first one. 0 Integer frequency (1 MHz) is used for f_{bdlc}. 1 Binary frequency (1.048576 MHz) is used for f_{bdlc}. Section 6.4.2.10, "J1850 VPW Valid/Invalid Bits and Symbols," describes the transmitter and receiver VPW symbol timing for integer and binary frequencies.
IE	Interrupt Enable This bit determines whether the BDLC module generates CPU interrupt requests. Interrupt requests are maintained until all of the interrupt request sources are cleared, by performing the specified actions upon the BDLC module's registers. Interrupts that were pending at the time that this bit is cleared may be lost. 0 Disable interrupt requests from BDLC module. 1 Enable interrupt requests from BDLC module. If the programmer does not wish to use the interrupt capability of the BDLC module, the BDLC State Vector Register can be polled periodically by the programmer to determine BDLC module states. Refer to for a description of Section 6.3.2.2, "BDLC State Vector Register (BDLC_DLCBSVR)," and how to clear interrupt requests.

6.3.2.2 BDLC State Vector Register (BDLC_DLCBSVR)

The read-only BDLC State Vector Register (BDLC_DLCBSVR) substantially decreases the CPU overhead associated with servicing interrupts while under operation of a MUX protocol. It provides an index offset directly related to the BDLC module's current state, which can be used with a user-supplied jump table to rapidly enter an interrupt service routine. This eliminates the need for the user to maintain a duplicate state machine in software.

Encoding interrupt sources in states allows that only one interrupt source has to be managed at a time. After the highest priority interrupt source is dealt with and if another interrupt event of a lower priority has also

occurred, the value corresponding to that interrupt source appears in the BDLC_DLCBSVR register. This continues until all BDLC interrupt sources have been managed and all bits in the BDLC_DLCBSVR register are cleared.

- Symbol invalid or out of range
- CRC error

  Cyclical Redundancy Check Byte is used by the receiver(s) of each message to determine if any errors have occurred during the transmission of the message. If the message is not error free, the CRC error status is shown in the BDLC_DLCBSVR register.
- Loss of arbitration

  Loss of arbitration status is entered when a loss of arbitration occurs while the BDLC is transmitting onto the bus.
- Tx data register empty

  The Tx data register empty (TDRE) state is used to tell when data has been unloaded from the BDLC Data Register.
- Rx data register full

  The Rx data register full (RDRF) state describes when data has been loaded in the BDLC Data Register.
- Received in-frame response (IFR) byte

  The BDLC can transmit and receive all four types of in-frame responses. As each byte of an IFR is received, the BDLC_DLCBSVR register indicates this by setting this state.
- Received EOF

  When a 280 µs passive period on the bus is received, it signifies an end-of-frame (EOF). When this occurs, the EOF flag is set.
- No interrupts pending

  This interrupt cannot generate an interrupt of the CPU.

Address: Base + 0x01 Access: User read-only

	0	1	2	3	4	5	6	7
R	0	0	I3	I2	I1	I0	0	0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-3. BDLC State Vector Register (BDLC_DLCBSVR)

Table 6-5. BDLC_DLCBSVR field descriptions

Field	Description
I[3:0]	Interrupt State Vector (with priority from low to high) 0000 No interrupt pending. 0001 Received EOF. 0010 Received IFR byte. 0011 Rx data register full. 0100 Tx data register empty. 0101 Loss of arbitration. 0110 CRC error. 0111 Symbol invalid or out of range. 1000 Reserved.

6.3.2.3 BDLC Control Register 2 (BDLC_DLCBCR2)

The BDLC Control Register 2 (BDLC_DLCBCR2) controls transmitter operations of the BDLC module.

Address: Base + 0x04 Access: User read/write

	0	1	2	3	4	5	6	7
R W	SMRST	DLOOP	4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
Reset	0	1	0	0	0	0	0	0

Figure 6-4. BDLC Control Register 2 (BDLC_DLCBCR2)

Table 6-6. BDLC_DLCBCR2 field descriptions

Field	Description
SMRST	State Machine Reset You can use this bit to reset the BDLC state machines to an initial state after the you put the off-chip analog transceiver in loop back mode. 0 Clearing SMRST after it has been set causes the generation of a state machine reset. After SMRST is cleared, the BDLC requires the bus to be idle for a minimum of an EOF symbol time before allowing the reception of a message. The BDLC requires the bus to be idle for a minimum of an inter-frame separator symbol (IFS) time before allowing any message to be transmitted. 1 Setting SMRST arms the state machine reset generation logic. Setting SMRST does not affect BDLC module behavior in any way.

Table 6-6. BDLC_DLCBCR2 field descriptions (Continued)

Field	Description
DLOOP	Digital Loopback Mode This bit determines the input source the digital filter is connected to and can be used to isolate bus fault conditions.If a fault condition has been detected on the bus, this control bit allows the programmer to disconnect the digital filter from input from the receive pin (RXB) and connect it to the transmit output to the pin (TXB). In this configuration, data sent from the transmit buffer should be reflected back into the receive buffer. If no faults exist in the digital block, the fault is in the physical interface block or elsewhere on the J1850 bus. 0 No loopback. When cleared, digital filter input is connected to receive pin (J1850_RX) and the transmitter output is connected to the transmit pin (J1850_TX). The BDLC module is taken out of Digital Loopback Mode and can now drive and receive from the J1850 bus normally. After writing DLOOP to zero, the BDLC module requires the bus to be idle for a minimum of an EOF symbol time before allowing a reception of a message. The BDLC module requires the bus to be idle for a minimum of an inter-frame separator symbol time before allowing any message to be transmitted. 1 Loopback. When set, digital filter input is connected to the transmitter output. The BDLC module is now in Digital Loopback Mode of operation. The transmit pin (J1850_TX) is driven low and not driven by the transmitter output. **Note:** The DLOOP bit is a fault condition aid and should never be altered after the BDLC Data Register is loaded for transmission. Changing DLOOP during a transmission may cause corrupted data to be transmitted onto the J1850 network.
4XE	4X Mode Enable This bit determines if the BDLC operates at normal transmit and receive speed (10.4 kbit/s) or in 4× mode at 41.6 kbit/s. This feature is useful for fast download of data into a J1850 node for diagnostic or factory programming of the node. The effect of 4× receive operation on receive symbol timing boundaries is described in Section 6.4.2.10, “J1850 VPW Valid/Invalid Bits and Symbols.” 0 When cleared, the BDLC module transmits and receives at 10.4 kbit/s. Reception of a BREAK symbol automatically clears this bit and sets the symbol invalid or out of range flag BDLC State Vector Register = 0x1C). 1 When set, the BDLC module is put in 4× (41.6 kbit/s) operation.
NBFS	Normalization Bit Format Select This bit controls the format of the normalization bit (NB). SAE J1850 strongly encourages the use of an active long: 0 for In-Frame Responses containing CRC and active short, 1 for In-Frame Responses without CRC. 0 NB that is received or transmitted is a 1 when the response part of an In-Frame Response (IFR) ends with a CRC byte. NB that is received or transmitted is a 0 when the response part of an In-Frame Response (IFR) does not end with a CRC byte. 1 NB that is received or transmitted is a 0 when the response part of an In-Frame Response (IFR) ends with a CRC byte. NB that is received or transmitted is a 1 when the response part of an In-Frame Response (IFR) does not end with a CRC byte.
TEOD	Transmit End of Data This bit is set by the programmer to indicate the end of a message being sent by the BDLC. It appends an 8-bit CRC after completing transmission of the current byte in the Tx Shift Register followed by the EOD symbol.If the transmit shadow register (refer to Section 6.4.4.1, “Protocol Architecture,” for a description of the transmit shadow register) is full when TEOD is set, the CRC byte and EOD is transmitted after the current byte in the Tx Shift Register and the byte in the Tx Shadow Register have been transmitted. Once TEOD is set, the transmit data register empty flag (TDRE) in the BDLC state vector register (BDLCSVR) is cleared to allow lower priority interrupts to occur. This bit is also used to end an IFR. Bits TSIFR, TMIFR1, and TMIFR0 determine whether a CRC byte is appended before EOD transmission for IFRs. 0 The TEOD bit is automatically cleared after the first CRC bit is sent, or if an error or loss of arbitration is detected on the bus. When TEOD is used to end an IFR transmission, TEOD is cleared when the BDLC receives back a valid EOD symbol, or an error condition or loss of arbitration occurs. 1 Transmit EOD symbol

Table 6-6. BDLC_DLCBCR2 field descriptions (Continued)

Field	Description
TSIFR	Transmit Single Byte IFR with no CRC (Type 1 or 2) 0 The TSIFR bit is automatically cleared after the EOD if one or more IFR bytes has been received or an error is detected on the bus. 1 If this bit is set prior to a valid EOD being received with no CRC error and after the EOD symbol has been received, the BDLC module attempts to transmit the appropriate normalization bit followed by the byte in the BDLC Data Register.
TMIFR1	Transmit Multiple Byte IFR with CRC (Type 3) 0 The TMIFR1 bit is automatically cleared once the BDLC module has successfully transmitted the CRC byte and EOD symbol, by the detection of an error on the multiplex bus, a transmitter underrun, or loss of arbitration. 1 If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received, the BDLC module attempts to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into BDLC Data Register. After TEOD has been set and the last IFR byte has been transmitted, the CRC byte is transmitted.
TMIFR0	Transmit Multiple Byte IFR with no CRC (Type 3) 0 The TMIFR0 bit is automatically cleared once the BDLC module has successfully transmitted the EOD symbol, by the detection of an error on the multiplex bus, a transmitter underrun, or loss of arbitration. 1 If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received the BDLC module attempts to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into BDLC Data Register. After TEOD has been set, the last IFR byte to be transmitted is the last byte written into the BDLC Data Register.

The TSIFR, TMIFR1, and TMIFR0 bits control the type of In-Frame Response being sent. The programmer should not set more than one of these control bits to 1 at any given time. However, if more than one of these three control bits are set to one, the priority encoding logic forces the internal register bits to a known value as shown in the following table. However, when these bits are read, they are the same as written earlier. For instance, if 011 is written to TSIFR, TMIFR1,and TMIFR0, then internally, they are encoded as 010. However, when these bits are later read back, they are encoded as 011.

The BDLC supports the In-frame Response (IFR) feature of J1850 by setting these bits correctly. The four types of J1850 IFR are shown in Figure 6-5. The purpose of the in-frame response modes is to allow single or multiple nodes to acknowledge receipt of the data by responding to a received message after they have seen the EOD symbol. For VPW modulation, the first bit of the IFR is always passive; therefore, an active normalization bit must be generated by the responder and sent prior to its ID/address byte. When there are multiple responders on the J1850 bus, only one normalization bit is sent, which assists all other transmitting nodes to sync their responses.

The TSIFR bit is used to request the BDLC to transmit the byte in the BDLC Data Register as a single byte IFR with no CRC. Typically, the byte transmitted is a unique identifier or address of the transmitting (responding) node.

Set the TSIFR bit before the EOF following the main part of the message frame is received or no IFR transmit attempts are made for the current message. If another node transmits an IFR to this message, set the TSIFR bit before the normalization bit is received or no IFR transmit attempts are made for the message. If another node does transmit a successful IFR or a reception error occurs, the TSIFR bit is cleared. If not, the IFR is transmitted after the EOD of the next received message.

If a loss of arbitration occurs when the BDLC module attempts transmission, after the IFR byte winning arbitration completes transmission, the BDLC module again attempts to transmit the byte in the BDLC Data Register (with no normalization bit). The BDLC module continues transmission attempts until an error is detected on the bus, or TEOD is set by the CPU, or the BDLC transmission is successful.

NOTE

Setting the TEOD bit before transmission of the IFR byte directs the BDLC to make only one attempt at transmitting the byte.

If loss of arbitration occurs in the last bit of the IFR byte, two additional 1 bits is not sent out because the BDLC attempts to retransmit the byte in the transmit shift register after the IFR byte winning arbitration completes transmission.

The TMIFR1 bit requests the BDLC module to transmit the byte in the BDLC Data Register (BDLC Data Register) as the first byte of a multiple byte IFR with CRC or as a single byte IFR with CRC. Response IFR bytes are subject to J1850 message length maximums.

After the byte in the BDLC Data Register has been loaded into the transmit shift register, the TDRE flag is set in the BDLC_DLCBSVR register, similar to the main message transmit sequence. If the interrupt enable bit (IE in the BDLC_DLCBCR1 register) is set, an interrupt request from the BDLC module is generated. The programmer should then load the next byte of the IFR into the BDLC Data Register for transmission. When the last byte of the IFR has been loaded into the BDLC Data Register, the programmer should set the TEOD bit in the BDLC control register 2. This instructs the BDLC module to transmit a CRC byte once the byte in the BDLC Data Register is transmitted, and then transmit an EOD symbol, indicating the end of the IFR portion of the message frame.

However, if you wish to transmit a single byte followed by a CRC byte, load the byte into the BDLC Data Register and then set the TMIFR1 bit before the EOD symbol has been received. Once the TDRE flag is set and interrupt occurs (if enabled), the programmer should then set the TEOD bit in BDLC Control Register 2. This results in the byte in the BDLC Data Register being the only byte transmitted before the IFR CRC byte.

Set the TMIFR1 bit before the EOF following the main part of the message frame is received or no IFR transmit attempts are made for the current message. If another node transmits an IFR to this message, set the TMIFR1 bit before the normalization bit is received or no IFR transmit attempts are made for the message. If another node does transmit a successful IFR or a reception error occurs, the TMIFR1 bit is cleared. If not, the IFR is transmitted after the EOD of the next received message.

If a transmitter underrun error occurs during transmission (caused by not writing another byte to the BDLC data register following the TDRE flag being set), the BDLC module automatically disables the transmitter after the byte currently in the shifter. Two extra 1-bits have been transmitted. The receiver picks this up as an framing error and relay it in the state vector register as an invalid symbol error. The TMIFR1 bit is also cleared.

If a loss of arbitration occurs when the BDLC module is transmitting a multiple byte IFR with CRC, the BDLC module goes to the loss of arbitration state, set the appropriate flag and cease transmission. The TMIFR1 bit is cleared and no attempt is made to retransmit the byte in the BDLC Data Register. If loss of arbitration occurs in the last bit of the IFR byte, two additional one bits (a passive long followed by an active short) is sent out.

NOTE

The extra logic is an enhancement to the J1850 protocol that forces a byte boundary condition fault. This helps prevent noise on the J1850 bus from corrupting a message.

The TMIFR0 bit is used to request the BDLC module to transmit the byte in the BDLC Data Register as the first byte of a multiple byte IFR without CRC. Response IFR bytes are subject to J1850 message length maximums.

After the byte in the BDLC Data Register has been loaded into the transmit shift register, the TDRE flag is set in the BDLC_DLCBSVR register, similar to the main message transmit sequence. If the interrupt enable bit (IE in the BDLC_DLCBCR1 register) is set, an interrupt request from the BDLC module is generated. The programmer should then load the next byte of the IFR into the BDLC Data Register for transmission. When the last byte of the IFR has been loaded into the BDLC Data Register, the programmer should set the TEOD bit in the BDLC Control Register 2. This instructs the BDLC to transmit an EOD symbol, indicating the end of the IFR portion of the message frame. The BDLC module does not append a CRC.

However, if the programmer wishes to transmit a single byte, the programmer should load the byte into the BDLC Data Register and then set the TMIFR0 bit before the EOD symbol has been received. Once the TDRE flag is set and interrupt occurs (if enabled), the programmer should then set the TEOD bit in BDLC Control Register 2. This results in the byte in the BDLC Data Register being the only byte transmitted.

Set the TMIFR0 bit before the EOF following the main part of the message frame is received, or no IFR transmit attempts is made for the current message. If another node transmits an IFR to this message, the user must set the TMIFR0 bit before the normalization bit is received or no IFR transmit attempts are made for the message. If another node does transmit a successful IFR or a reception error occurs, the TMIFR0 bit is cleared. If not, the IFR is transmitted after the EOD of the next received message.

If a transmitter underrun error occurs during transmission (caused by the programmer not writing another byte to the BDLC Data Register following the TDRE flag being set) the BDLC module automatically disables the transmitter after the byte currently in the shifter plus two extra 1-bits have been transmitted. The receiver picks this up as an framing error and relay it in the State Vector Register as an invalid symbol error. The TMIFR0 bit is also cleared.

If a loss of arbitration occurs when the BDLC module is transmitting a multiple byte IFR without CRC, the BDLC module goes to the loss of arbitration state, set the appropriate flag and cease transmission. The TMIFR0 bit is cleared and no attempt is made to retransmit the byte in the BDLC Data Register. If loss of arbitration occurs in the last bit of the IFR byte, two additional one bits (a passive long followed by an active short) is sent out.

NOTE

The extra logic is an enhancement to the J1850 protocol that forces a byte boundary condition fault. This helps prevent noise on the J1850 bus from corrupting a message.

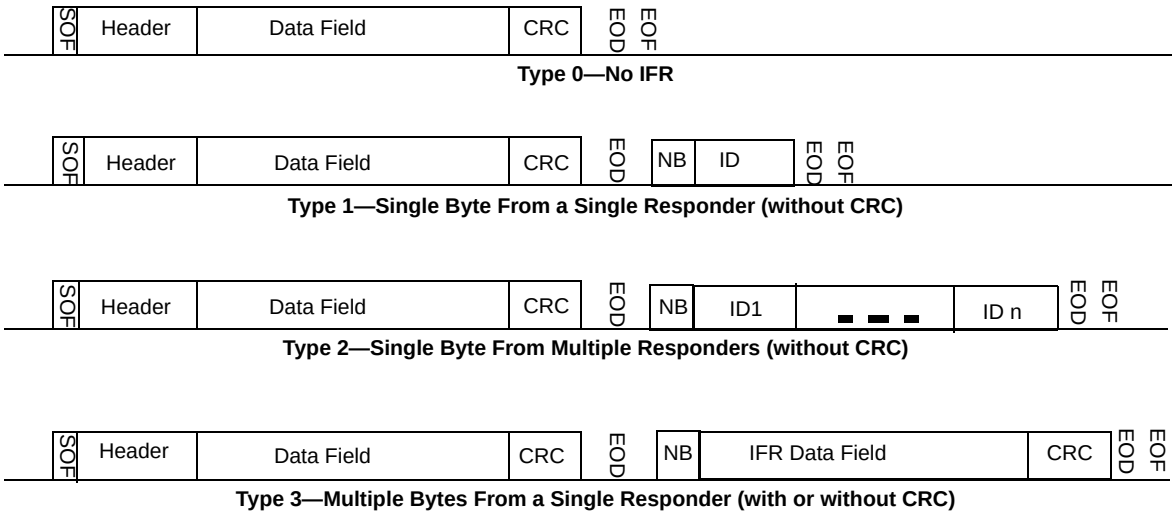


Figure 6-5. Types of In-Frame Response

Table 6-7. Transmit In-Frame Response Control Bit Priority Encoding

WRITE			READ			ACTUAL (Internal Register)		
TSIFR	TMIFR1	TMIFR0	TSIFR	TMIFR1	TMIFR0	TSIFR	TMIFR1	TMIFR0
0	0	0	0	0	0	0	0	0
1	—	—	1	—	—	1	0	0
0	1	—	0	1	—	0	1	0
0	0	1	0	0	1	0	0	1

6.3.2.4 BDLC Data Register (BDLC_DLCBDR)

The BDLC Data Register (BDLC_DLCBDR) passes the data to be transmitted to the J1850 bus from the CPU to the BDLC module. It is also used to pass data received from the J1850 bus to the CPU.

While transmitting, each data byte (after the first one) should be written only after a Tx Data Register Empty (TDRE) interrupt has occurred, or the BDLC_DLCBSVR register has been polled indicating this condition.

Data read from this register is the last data byte received from the J1850 bus. This received data should only be read after a Rx Data Register Full (RDRF) or Received IFR byte (RXIFR) interrupt has occurred or the BDLC_DLCBSVR register has been polled indicating either of these two conditions.

The BDLC Data Register is double buffered via a transmit shadow register and a receive shadow register. After the byte in the transmit shift register has been transmitted, the byte currently stored in the transmit shadow register is loaded into the transmit shift register. Once the transmit shift register has shifted the first bit out, the TDRE flag is set, and the shadow register is ready to accept the next byte of data.

The receive shadow register works similarly. Once a complete byte has been received, the receive shift register stores the newly received byte into the receive shadow register. The RDRF flag (or RXIFR flag if the received byte is part of an IFR) is set to indicate that a new byte of data has been received. The programmer has one BDLC module byte reception time to read the shadow register and clear the RDRF or RXIFR flag before the shadow register is overwritten by the newly received byte.

If the user writes the first byte of a message to be transmitted to the BDLC Data Register and then determines that a different message should be transmitted, the user can write a new byte to the BDLC Data Register up until the transmission begins. This new byte replaces the original byte in the BDLC Data Register.

From the time a byte is written to the BDLC Data Register until it is transferred to the transmit shift register, the transmit shadow register is considered full and the byte pending transmission. If one of the IFR transmission control bits (TSIFR, TMIFR1, or TMIFR0 in BDLC Control Register 2) is also set, the byte is pending transmission as an IFR. A byte pending transmission is flushed from the transmit shadow register and the transmission canceled if one of the following occurs: a loss of arbitration or transmitter error on the byte currently being transmitted; a symbol error, framing error, bus fault, or BREAK symbol is received. If the byte pending transmission is an IFR byte, the reception of a message with a CRC error also causes the byte in the transmit shadow register to be flushed.

To abort an in-progress transmission, the programmer should simply stop loading more data into the BDLC Data Register. This causes a transmitter underrun error and the BDLC module automatically disables the transmitter on the next non-byte boundary. This means that the earliest a transmission can be halted is after at least one byte (plus two extra 1-bits) has been transmitted. The receiver picks this up as an error and relays it in the state vector register as an invalid symbol error.

Address: Base + 0x05 Access: User read/write

	0	1	2	3	4	5	6	7
R W	D7	D6	D5	D4	D3	D2	D1	D0
Reset	0	0	0	0	0	0	0	0

Figure 6-6. BDLC Data Register (BDLC_DLCBDR)

Table 6-8. DLCBDR field descriptions

Field	Description
D[7:0]	Receive/Transmit Data

6.3.2.5 BDLC Analog Round Trip Delay Register (BDLC_DLCBARD)

The BDLC Analog Round Trip Delay Register (BDLC_DLCBARD) is used to program the BDLC module so that it compensates for the round trip delays of different external transceivers. Also the polarity of the receive pin (J1850_RX) is set in this register.

Read: any time

Write: write only once.

Address: Base + 0x08 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	RXPOL	0	BO4	BO3	BO2	BO1	BO0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-7. BDLC Analog Round Trip Delay Register (BDLC_DLCBARD)

Table 6-9. BDLC_DLCBARD field descriptions

Field	Description
RXPOL	Receive Pin Polarity The Receive pin Polarity bit is used to select the polarity of incoming signal on the receive pin. Some external analog transceiver inverts the receive signal from the J1850 bus before feeding back to the digital receive pin. 0 Select inverted polarity, where external transceiver inverts the receive signal. 1 Select normal/true polarity; true non-inverted signal from J1850 bus, i.e., the external transceiver does not invert the receive signal.
BO[4:0[	BDLC Analog Roundtrip Delay Offset Field Adjust the transmitted symbol timings to account for the differing round-trip delays found in different SAE J1850 analog transceivers.The allowable delay range is from 0 ms to 31 ms, with a nominal target of 16 ms (reset value). Refer to Table 6-10 for the BO[4:0] values corresponding to the expected transceiver delays and the resultant transmitter timing adjustment (in MUX interface clock periods (t_{bdlc})). Refer to the analog transceiver device specification for the expected round-trip delay through both the transmitter and the receiver. The sum of these two delays makes up the total round-trip delay value. **Note:** For Digital Loopback test, the Analog Round-trip Delay Offset Field should be set to 0 µs.

Table 6-10. BARD Values vs. Transceiver Delay and Transmitter Timing Adjustment

BARD Offset Bits BO[4:0]	Corresponding Expected Transceiver's delays (µs)	Transmitter Symbol Timing Adjustment (t_{bdlc}[1])
00000	0	0
00001	1	1
00010	2	2
00011	3	3
00100	4	4
00101	5	5
00110	6	6
00111	7	7
01000	8	8
01001	9	9
01010	10	10
01011	11	11
01100	12	12
01101	13	13
01110	14	14

Table 6-10. BARD Values vs. Transceiver Delay and Transmitter Timing Adjustment (Continued)

BARD Offset Bits BO[4:0]	Corresponding Expected Transceiver's delays (µs)	Transmitter Symbol Timing Adjustment (t_{bdlc}[1])
01111	15	15
10000	16	16
10001	17	17
10010	18	18
10011	19	19
10100	20	20
10101	21	21
10110	22	22
10111	23	23
11000	24	24
11001	25	25
11010	26	26
11011	27	27
11100	28	28
11101	29	29
11110	30	30
11111	31	31

1 The transmitter symbol timing adjustment is the same for binary and integer bus frequencies.

6.3.2.6 BDLC Rate Select Register (BDLC_DLCBRSR)

The BDLC Rate Select Register (BDLC_DLCBRSR) determines the divider prescaler value for the MUX interface clock (f_{bdlc}). Only integer multiples of the 1 MHz or 1.048576 MHz f_{bdlc} are supported as input clocks.

Read: any time

Write: write only once.

Address: Base + 0x09 Access: User read/write

	0	1	2	3	4	5	6	7
R W	R7	R6	R5	R4	R3	R2	R1	R0
Reset	0	0	0	0	0	0	0	0

Figure 6-8. BDLC Rate Select Register (BDLC_DLCBRSR)

Table 6-11. BDLC_DLCBRSR field descriptions

Field	Description
R[7:0]	Rate Select. These bits determine the amount by which the frequency of the system clock signal is divided to generate the MUX Interface clock (f_{bdlc}), which defines the basic timing resolution of the MUX Interface.The value programmed into these bits depends on the chosen system clock frequency. See Table 6-12 and Table 6-13 for example rate selects for different bus frequencies. All divisor values from divide by 1 to divide by 256 are possible, but are not shown in the tables. **Note:** Although the maximum divider is 256, a divider that generates a 1 MHz or 1.048576 MHz f_{bdlc} must be selected for J1850 communications to occur.

Table 6-12. BDLC Rate Selection for Binary Frequencies [CLKS = 1]

IP bus clock frequency	R[7:0]	division	f_{bdlc}
f_{CLOCK} = 1.048576 MHz	0x00	1	1.048576 MHz

Table 6-13. BDLC Rate Selection for Binary Frequencies [CLKS = 1]

IP bus clock frequency	R[7:0]	division	f_{bdlc}
f_{CLOCK} = 66.00000 MHz	0x41	66	1.000000 MHz
f_{CLOCK} = 54.00000 MHz	0x35	54	1.000000 MHz
f_{CLOCK} = 33.00000 MHz	0x20	33	1.000000 MHz
f_{CLOCK} = 27.00000 MHz	0x1A	27	1.000000 MHz

6.3.2.7 BDLC Control Register (BDLC_DLCSCR)

The BDLC Control Register (BDLC_DLCSCR) enables the BDLC module.

Address: Base + 0x0C Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	BDLCE	0	0	0	BREAK
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-9. BDLC Control Register (BDLC_DLCSCR)

Table 6-14. BDLC_DLCSCR field descriptions

Field	Description
BDLCE	BDLC Enable This bit serves as a MUX interface clock (f_{bdlc}) enable/disable for power savings. 0 The MUX interface clock (f_{bdlc}) is disabled, shutting down the BDLC module for power saving. Bus clocks continue running, allowing registers to be accessed. 1 The MUX interface clock (f_{bdlc}) and BDLC module are enabled to allow J1850 communications to take place.
BREAK	Send BREAK signal This bit determines whether the BDLC module generates a BREAK symbol. 0 The BDLC module does not generate a BREAK symbol. 1 The BDLC module immediately sends a Break signal on the bus, regardless of its current transmit or receive status. After setting the BREAK bit it is automatically cleared after two IPB clock cycles. The active Break signal causes any other transmitting module to stop transmitting immediately because it loses arbitration. It is at least 280 µs long. **Note:** When the BDLC is operating at the high bus speed all 4X symbol times are one fourth that shown, except for Break, which is transmitted the same length in 1X or 4X mode.

6.3.2.8 BDLC Status Register (BDLC_DLCBSTAT)

The BDLC Status Register (BDLC_DLCBSTAT) indicates the status of the BLDC module.

Read: any time

Write: any time

Address: Base + 0x0D Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	TST_DIVTE_T4	TST_CRCV_T4	IDLE
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-10. BDLC Status Register (BDLC_DLCBSTAT)

Table 6-15. BDLC_DLCBSTAT field descriptions

Field	Description
TST_DIVTE_T4	. 0 The module output Tx pin output normal signal. 1 The module output Tx pin output MUX interface clock (FBDLC).
TST_CRCV_T4	Status of the receive message CRC. 0 Correct. 1 Incorrect.
IDLE	This bit indicates when the BDLC module is idle. 0 BDLC module is either transmitting or receiving data. 1) BDLC module has received IFS and no data is being transmitted or received. **Note:** BDLC module is only idle after receiving IFS. The IDLE bit is 0 during reset since the BDLC module needs to wait for an IFS before becoming idle. Noise on the bus will be filtered and the IDLE bit will remain unchanged.

6.4 Functional Description

The BDLC module allows the user to send and receive messages across an SAE J1850 serial communication network. User software manages each transmitted or received message on a byte-by-byte basis, while the BDLC performs all of the network access, arbitration, message framing and error detection duties.

6.4.1 J1850 Frame Format

As noted above and in Section 6.1.1, “Features,” the BDLC module communicates across an SAE J1850 network. As such, all messages transmitted on the J1850 bus are structured using the format below. The following sections describe this format and its meanings.

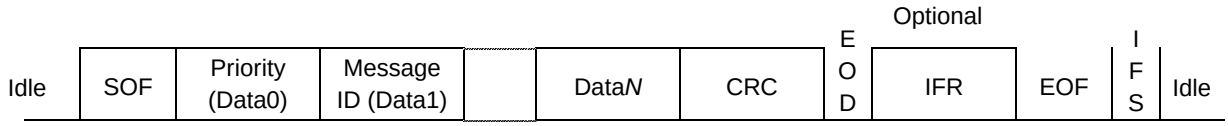


Figure 6-11. J1850 Bus Message Format (VPW)

SAE J1850 states that each message has a maximum length of 101 bit times or 12 bytes (excluding SOF, EOD, NB, and EOF).

6.4.1.1 Start of Frame Symbol (SOF)

All messages transmitted onto the J1850 bus must begin with an long active SOF symbol. This indicates to any listeners on the J1850 bus the start of a new message transmission. The SOF symbol is not used in the CRC calculation.

6.4.1.2 In Message Data Bytes (Data)

The data bytes contained in the message include the message priority/type, message I.D. byte, and any actual data being transmitted to the receiving node. See SAE J1850 - Class B Data Communications Network Interface, for more information about 1 and 3 Byte Headers.

Messages transmitted by the BDLC module onto the J1850 bus must contain at least one data byte, and therefore can be as short as one data byte and one CRC byte. Each data byte in the message is 8 bits in length, transmitted MSB to LSB.

6.4.1.3 Cyclical Redundancy Check Byte (CRC)

This byte is used by the receiver(s) of each message to determine if any errors have occurred during the transmission of the message. The BDLC calculates the CRC byte and appends it onto any messages transmitted onto the J1850 bus, and also performs CRC detection on any messages it receives from the J1850 bus.

CRC generation uses the divisor polynomial $X^8 + X^4 + X^3 + X^2 + 1$. The remainder polynomial is initially set to all ones, and then each byte in the message after the SOF symbol is serially processed through the CRC generation circuitry. The one's complement of the remainder then becomes the 8-bit CRC byte, which is appended to the message after the data bytes, in MSB to LSB order.

When receiving a message, the BDLC uses the same divisor polynomial. All data bytes, excluding the SOF and EOD symbols, but including the CRC byte, are used to check the CRC. If the message is error free, the remainder polynomial equals $X^7 + X^6 + X^2$ (0xC4), regardless of the data contained in the message. If the calculated CRC does not equal 0xC4, the BDLC recognizes this as a CRC error and set the CRC error flag in the BDLC_DLCBSVR register.

6.4.1.4 End-of-Data Symbol (EOD)

The EOD symbol is a long passive period on the J1850 bus used to signify to any recipients of a message that the transmission by the originator has completed. No flag is set upon reception of the EOD symbol.

6.4.1.5 In-Frame Response Bytes (IFR)

The IFR section of the J1850 message format is optional. Users desiring further definition of in-frame response should review the SAE J1850 Class B Data Communications Network Interface specification.

6.4.1.6 End-of-Frame Symbol (EOF)

This symbol is a passive period on the J1850 bus, longer than an EOD symbol, which signifies the end of a message. Because an EOF symbol is longer than an EOD symbol, if no response is transmitted after an EOD symbol, it becomes an EOF, and the message is assumed to be completed. The EOF flag is set upon receiving the EOF symbol.

6.4.1.7 Inter-Frame Separation Symbol (IFS)

The IFS symbol is a passive period on the J1850 bus that allows proper synchronization between nodes during continuous message transmission. The IFS symbol is transmitted by a node following the completion of the EOF period.

When the last byte of a message has been transmitted onto the J1850 bus, and the EOF symbol time has expired, all nodes must then wait for the IFS symbol time to expire before transmitting an SOF, marking the beginning of another message.

However, if the BDLC module is waiting for the IFS period to expire before beginning a transmission and a rising edge is detected before the IFS time has expired, it internally synchronizes to that edge.

A rising edge may occur during the IFS period because of varying clock tolerances and loading of the J1850 bus, causing different nodes to observe the completion of the IFS period at different times. Receivers must synchronize to any SOF occurring during an IFS period to allow for individual clock tolerances.

6.4.1.8 Break

If the BDLC module is transmitting at the time a BREAK is detected, it treats the BREAK as if a transmission error had occurred, and halts transmission.The BDLC module can transmit a BREAK symbol. If while receiving a message the BDLC module detects a BREAK symbol, it treats the BREAK as a reception error and sets the invalid symbol flag. If while receiving a message in 4X mode, the BDLC module detects a BREAK symbol, it treats the BREAK as a reception error, sets BDLC_DLCBSVR register to 0x1C, and exits 4X mode.The 4XE bit in BDLC Control Register 2 is automatically cleared upon reception of the BREAK symbol.

6.4.1.9 Idle Bus

An idle condition exists on the bus during any passive period after expiration of the IFS period. Any node sensing an idle bus condition can begin transmission immediately.

6.4.2 J1850 VPW Symbols

Variable pulse width modulation (VPW) is an encoding technique in which each bit is defined by the time between successive transitions, and by the level of the bus between transitions, active or passive. Active and passive bits are used alternately. This encoding technique is used to reduced the number of bus transitions for a given bit rate. See Section 6.1.1, “Features.”

The symbol values shown below are nominal values. Refer to the electrical specification for a more complete description of symbol values. Each logic one or logic zero contains a single transition, and can be at either the active or passive level and one of two lengths, either 64 µs or 128 µs (T_{NOM} at 10.4 kbit/s baud rate), depending upon the encoding of the previous bit. The SOF, EOD, EOF and IFS symbols are always encoded at an assigned level and length. See Figure 6-12.

Each message begins with an SOF symbol, an active symbol. Therefore, each data byte (including the CRC byte) begins with a passive bit, regardless of whether it is a logic 1 or a logic 0. All VPW bit lengths stated in the following descriptions are typical values at a 10.4 kbit/s bit rate.

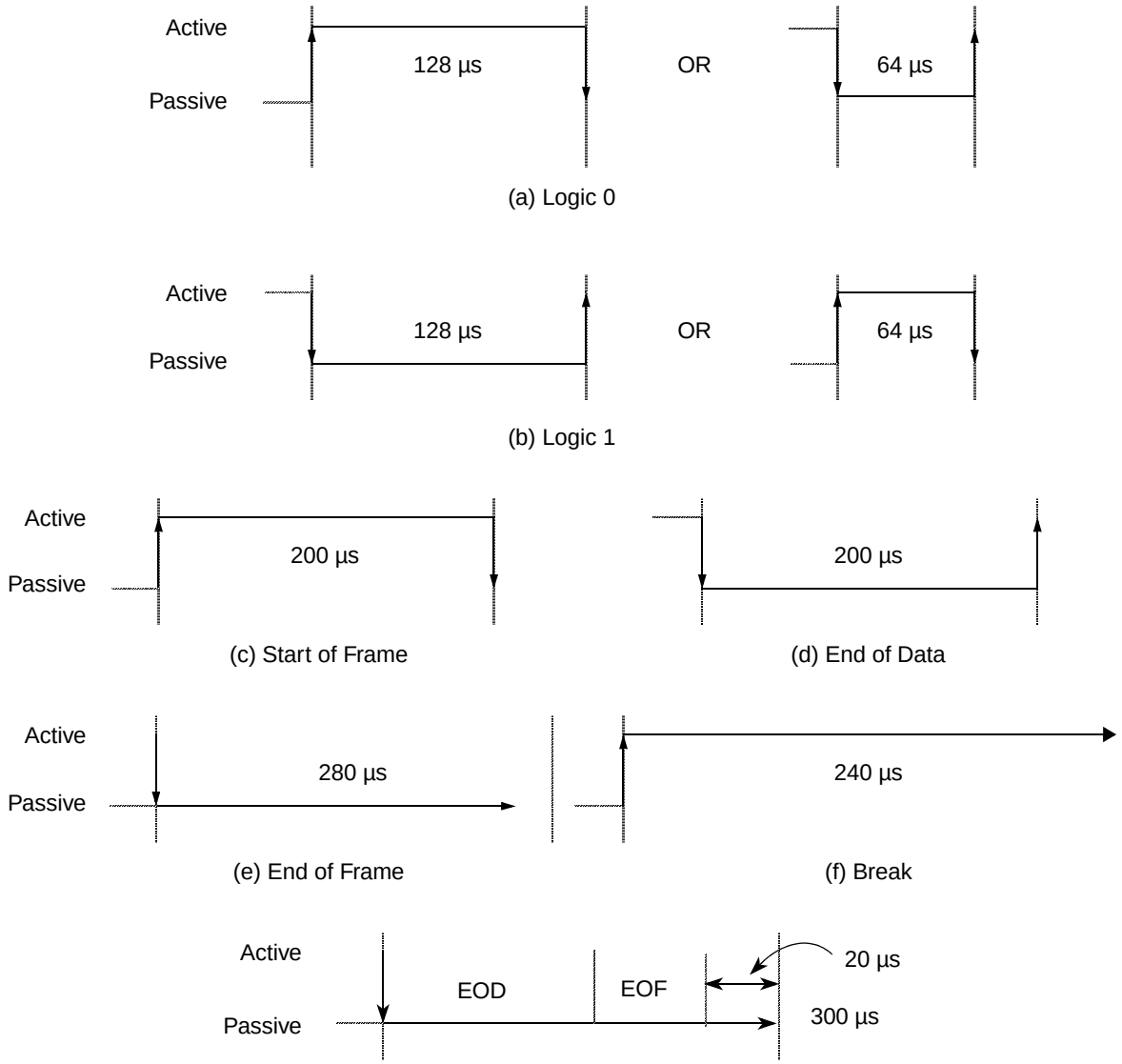


Figure 6-12. J1850 VPW Symbols

6.4.2.1 Logic 0

A logic zero is defined as either an active to passive transition followed by a passive period 64 µs in length, or a passive to active transition followed by an active period 128 µs in length (see Figure 6-12(a)).

6.4.2.2 Logic 1

A logic one is defined as either an active to passive transition followed by a passive period 128 µs in length, or a passive to active transition followed by an active period 64 µs in length (see Figure 6-12 (b)).

6.4.2.3 Normalization Bit (NB)

The NB symbol has the same property as a logic 1 or a logic 0. It is only used in IFR message responses. This bit is defined as an active bit.

6.4.2.4 Start of Frame Symbol (SOF)

The SOF symbol is defined as passive to active transition followed by an active period 200 µs in length (see Figure 6-12 (c)). This allows the data bytes that follow the SOF symbol to begin with a passive bit, regardless of whether it is a logic one or a logic zero.

6.4.2.5 End of Data Symbol (EOD)

The EOD symbol is defined as an active to passive transition followed by a passive period 200 µs in length (see Figure 6-12 (d)).

6.4.2.6 End of Frame Symbol (EOF)

The EOF symbol is defined as an active to passive transition followed by a passive period 280 µs in length (see Figure 6-12 (e)). If there is no IFR byte transmitted after an EOD symbol is transmitted, after another 80 µs the EOD becomes an EOF, indicating the completion of the message.

6.4.2.7 Inter-Frame Separation Symbol (IFS)

The IFS symbol is defined as a passive period 300 µs in length. The IFS symbol contains no transition, since when used it always follows an EOF symbol.(see Figure 6-12 (g))

6.4.2.8 Break Signal (BREAK)

The BREAK signal is defined as a passive to active transition followed by an active period of at least 240 µs (see Figure 6-12 (f)).

6.4.2.9 IDLE

An IDLE is defined as a passive period greater than 3000 µs in length.

6.4.2.10 J1850 VPW Valid/Invalid Bits and Symbols

The timing tolerances for receiving data bits and symbols from the J1850 bus have been defined to allow for variations in oscillator frequencies. In many cases the maximum time allowed to define a data bit or symbol is equal to the minimum time allowed to define another data bit or symbol.

Because the minimum resolution of the BDLC module for determining which symbol is received equals a single period of the MUX Interface clock (t_{bdlc}), the receiver symbol timing boundaries are subject to an uncertainty of 1 t_{bdlc} due to sampling considerations.

This clock resolution of 1 t_{bdlc} allows the BDLC module to properly differentiate between the different bits and symbols, without reducing the valid window for receiving bits and symbols from transmitters onto the J1850 bus having varying oscillator frequencies.

6.4.2.10.1 Transmit and Receive Symbol Timing Specifications

Table 6-16 through Table 6-21 contain the SAE J1850 transmit and receive symbol timing specifications for the BDLC module. The units used in these tables are MUX interface clock periods (t_{bdlc}). The MUX interface clock is a divided down version of the bus clock input to the module (see Section 6.3.2.6, "BDLC Rate Select Register (BDLC_DLCBRSR)"). The MUX interface clock drives the transmit and receive counters that control symbol generation and identification. The symbol timing in effect during J1850 operations depends on the state of two control bits: the CLKS bit in the BDLC_DLCBCR1 register, which indicates whether the bus clock is an integer frequency or a binary frequency; the 4XE bit in BDLC Control Register 2, which is used to select 4X operation.

Table 6-16 andTable 6-18 indicate the transmit and receive timing for integer bus frequencies (CLKS = 0) and 4X operation disabled (4XE = 0). It is assumed that for integer bus frequencies the divided down MUX interface clock frequency is 1 MHz (t_{bdlc} = 10 µs).

Table 6-17 and Table 6-19 indicated the transmit and receive timing for binary bus frequencies (CLKS = 1) and 4X operation disabled (4XE = 0). It is assumed that the divided down MUX interface clock frequency is 1.048576 MHz (t_{bdlc} = 0.953674 0 µs) for binary bus frequencies. The symbol timing values are adjusted to compensate for the shortening of the MUX interface clock period.

Table 6-20 and Table 6-21 show how the receive symbol timing values are adjusted when 4X operation is enabled (4XE = 1) for both integer bus frequencies (CLKS = 0) and binary bus frequencies (CLKS = 1), respectively.

The values specified in the tables are for the symbols appearing on the SAE J1850 bus. These values assume the BDLC module is communicating on the SAE J1850 bus using an external analog transceiver, and that the BDLC module analog round-trip delay value programmed into the BDLC_DLCBARD register is the appropriate value for the transceiver being used. If these conditions are not met, the symbol timings being measured on the SAE J1850 bus are significantly affected. For a detailed description of how symbol timings are measured on the SAE J1850 bus, refer to the appropriate SAE documents.

Table 6-16. BDLC Transmitter VPW Symbol Timing for Integer Frequencies

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	T_{tvp1}	62	64	66	t_{bdlc}
2	Passive Logic 1	T_{tvp2}	126	128	130	t_{bdlc}
3	Active Logic 0	T_{tva1}	126	128	130	t_{bdlc}
4	Active Logic 1	T_{tva2}	62	64	66	t_{bdlc}
5	Start of Frame (SOF)	T_{tva3}	198	200	202	t_{bdlc}
6	End of Data (EOD)[1]	T_{tvp3}	162	164	166	t_{bdlc}
7	End of Frame (EOF)[1]	T_{tv4}	238	240	242	t_{bdlc}
8	Inter-Frame Separator (IFS)[1]	T_{tv5}	298	300	302	t_{bdlc}

Note: The transmitter timing for this symbol depends upon the minimum detection time of the symbol by the receiver.

Table 6-17. BDLC Transmitter VPW Symbol Timing for Binary Frequencies

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	T_{tvp1}	65	67	69	t_{bdlc}
2	Passive Logic 1	T_{tvp2}	132	134	136	t_{bdlc}
3	Active Logic 0	T_{tva1}	132	134	136	t_{bdlc}
4	Active Logic 1	T_{tva2}	65	67	69	t_{bdlc}
5	Start of Frame (SOF)	T_{tva3}	208	210	212	t_{bdlc}
6	End of Data (EOD)[1]	T_{tvp3}	170	172	174	t_{bdlc}
7	End of Frame (EOF)[1]	T_{tv4}	250	252	254	t_{bdlc}
8	Inter-Frame Separator (IFS)[1]	T_{tv5}	313	315	317	t_{bdlc}

[1] The transmitter timing for this symbol depends upon the minimum detection time of the symbol by the receiver.

Table 6-18. BDLC Receiver VPW Symbol Timing for Integer Frequencies

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	T_{rvp1}	32	64	95	t_{bdlc}
2	Passive Logic 1	T_{rvp2}	96	128	163	t_{bdlc}
3	Active Logic 0	T_{rva1}	96	128	163	t_{bdlc}
4	Active Logic 1	T_{rva2}	32	64	95	t_{bdlc}
5	Start of Frame (SOF)	T_{rva3}	164	200	239	t_{bdlc}
6	End of Data (EOD)	T_{rvp3}	164	200	239	t_{bdlc}
7	End of Frame (EOF)	T_{rv4}	240	280	299	t_{bdlc}
8	Inter-Frame Separator (IFS)	T_{rv5}	281	—	—	t_{bdlc}
9	Break Signal (BREAK)	T_{rv6}	240	—	—	t_{bdlc}

Note: The receiver symbol timing boundaries are subject to an uncertainty of 1 t_{bdlc} due to sampling considerations.

Table 6-19. BDLC Receiver VPW Symbol Timing for Binary Frequencies

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	T_{rvp1}	34	67	100	t_{bdlc}
2	Passive Logic 1	T_{rvp2}	101	134	171	t_{bdlc}
3	Active Logic 0	T_{rva1}	101	134	171	t_{bdlc}
4	Active Logic 1	T_{rva2}	34	67	100	t_{bdlc}
5	Start of Frame (SOF)	T_{rva3}	172	210	251	t_{bdlc}
6	End of Data (EOD)	T_{rvp3}	172	210	251	t_{bdlc}
7	End of Frame (EOF)	T_{rv4}	252	293	314	t_{bdlc}
8	Inter-Frame Separator (IFS)	T_{rv5}	315	—	—	t_{bdlc}
9	Break Signal (BREAK)	T_{rv6}	252	—	—	t_{bdlc}

Note: The receiver symbol timing boundaries are subject to an uncertainty of 1 t_{bdlc} due to sampling considerations.

Table 6-20. BDLC Receiver VPW 4X Symbol Timing for Integer Frequencies

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	T_{rvp1}	8	16	23	t_{bdlc}
2	Passive Logic 1	T_{rvp2}	24	32	40	t_{bdlc}
3	Active Logic 0	T_{rva1}	24	32	40	t_{bdlc}
4	Active Logic 1	T_{rva2}	8	16	23	t_{bdlc}
5	Start of Frame (SOF)	T_{rva3}	41	50	59	t_{bdlc}
6	End of Data (EOD)	T_{rvp3}	41	50	59	t_{bdlc}
7	End of Frame (EOF)	T_{rv4}	60	70	74	t_{bdlc}
8	Inter-Frame Separator (IFS)	T_{rv5}	75	—	—	t_{bdlc}
9	Break Signal (BREAK)	T_{rv6}	60	—	—	t_{bdlc}

Note: The receiver symbol timing boundaries are subject to an uncertainty of 1 t_{bdlc} due to sampling considerations.

Table 6-21. BDLC Receiver VPW 4X Symbol Timing for Binary Frequencies

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	T_{rvp1}	9	17	25	t_{bdlc}
2	Passive Logic 1	T_{rvp2}	26	34	42	t_{bdlc}
3	Active Logic 0	T_{rva1}	26	34	42	t_{bdlc}
4	Active Logic 1	T_{rva2}	9	17	25	t_{bdlc}
5	Start of Frame (SOF)	T_{rva3}	43	53	62	t_{bdlc}
6	End of Data (EOD)	T_{rvp3}	43	53	62	t_{bdlc}
7	End of Frame (EOF)	T_{rv4}	63	74	78	t_{bdlc}
8	Inter-Frame Separator (IFS)	T_{rv5}	79	—	—	t_{bdlc}
9	Break Signal (BREAK)	T_{rv6}	63	—	—	t_{bdlc}

Note: The receiver symbol timing boundaries are subject to an uncertainty of 1 t_{bdlc} due to sampling considerations.

The minimum and maximum symbol limits shown in the following sections (Invalid Passive Bit–Valid BREAK Symbol) and figures (Figure 6-13 through Figure 6-16) refer to the values listed in Table 6-16 through Table 6-21.

Invalid Passive Bit

If the passive to active transition beginning the next data bit or symbol occurs between the active to passive transition beginning the current data bit or symbol and $T_{rvp1(Min)}$, the current bit would be invalid. See Figure 6-13(1).

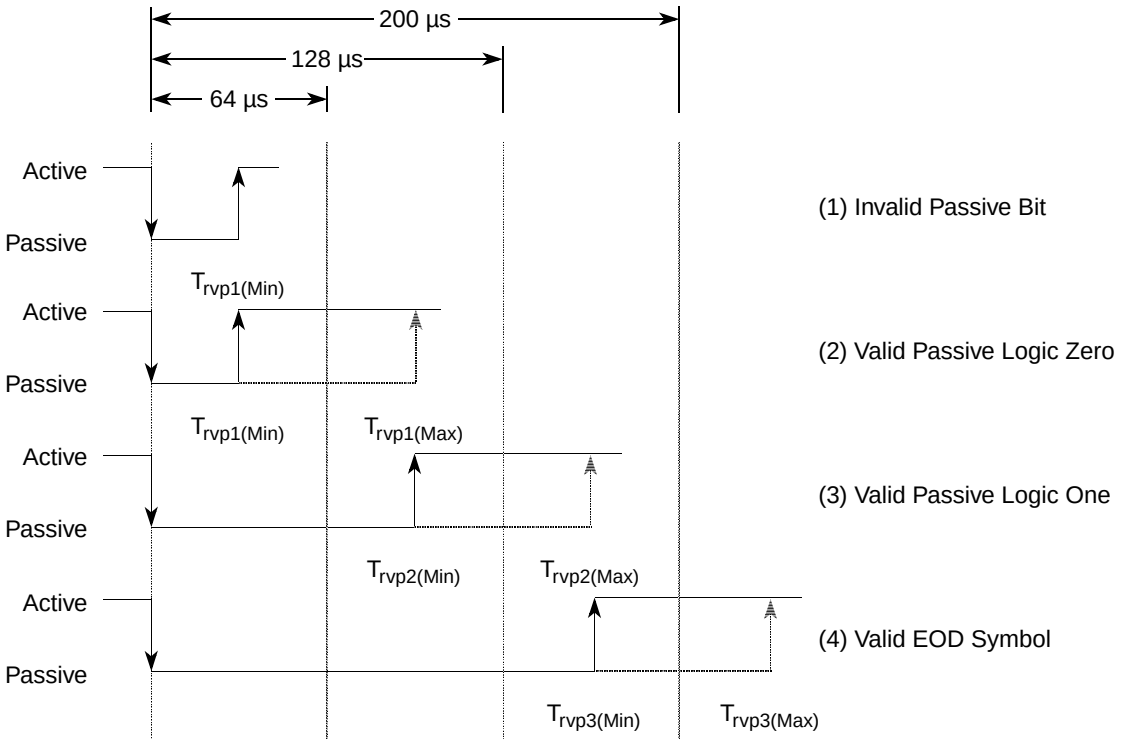


Figure 6-13. J1850 VPW Passive Symbols

Valid Passive Logic Zero

If the passive to active transition beginning the next data bit or symbol occurs between $T_{rvp1(Min)}$ and $T_{rvp1(Max)}$, the current bit would be considered a logic zero. See Figure 6-13(2).

Valid Passive Logic One

If the passive to active transition beginning the next data bit or symbol occurs between $T_{rvp2(Min)}$ and $T_{rvp2(Max)}$, the current bit would be considered a logic one. See Figure 6-13(3).

Valid EOD Symbol

If the passive to active transition beginning the next data bit or symbol occurs between $T_{rvp3(Min)}$ and $T_{rvp3(Max)}$, the current symbol would be considered a valid EOD symbol. See Figure 6-13(4).

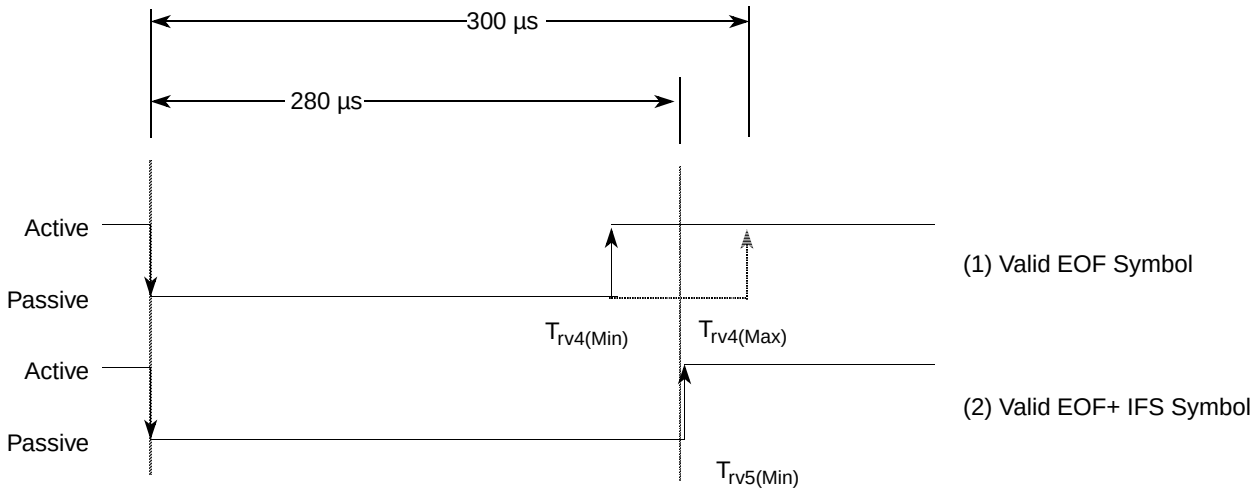


Figure 6-14. J1850 VPW EOF and IFS Symbols

Valid EOF and IFS Symbol

In Figure 6-14(1), if the passive to active transition beginning the SOF symbol of the next message occurs between $T_{rv4(Min)}$ and $T_{rv4(Max)}$, the current symbol is considered a valid EOF symbol.

If the passive to active transition beginning the SOF symbol of the next message occurs after $T_{rv5(Min)}$, the current symbol is considered a valid EOF symbol followed by a valid IFS symbol. See Figure 6-14(2). All nodes must wait until a valid IFS symbol time has expired before beginning transmission. However, due to variations in clock frequencies and bus loading, some nodes may recognize a valid IFS symbol before others, and immediately begin transmitting. Therefore, anytime a node waiting to transmit detects a passive to active transition once a valid EOF has been detected, it should immediately begin transmission, initiating the arbitration process.

Idle Bus

If the passive to active transition beginning the SOF symbol of the next message does not occur before $T_{tv5(Min)}$, the bus is considered to be idle, and any node wishing to transmit a message may do so immediately.

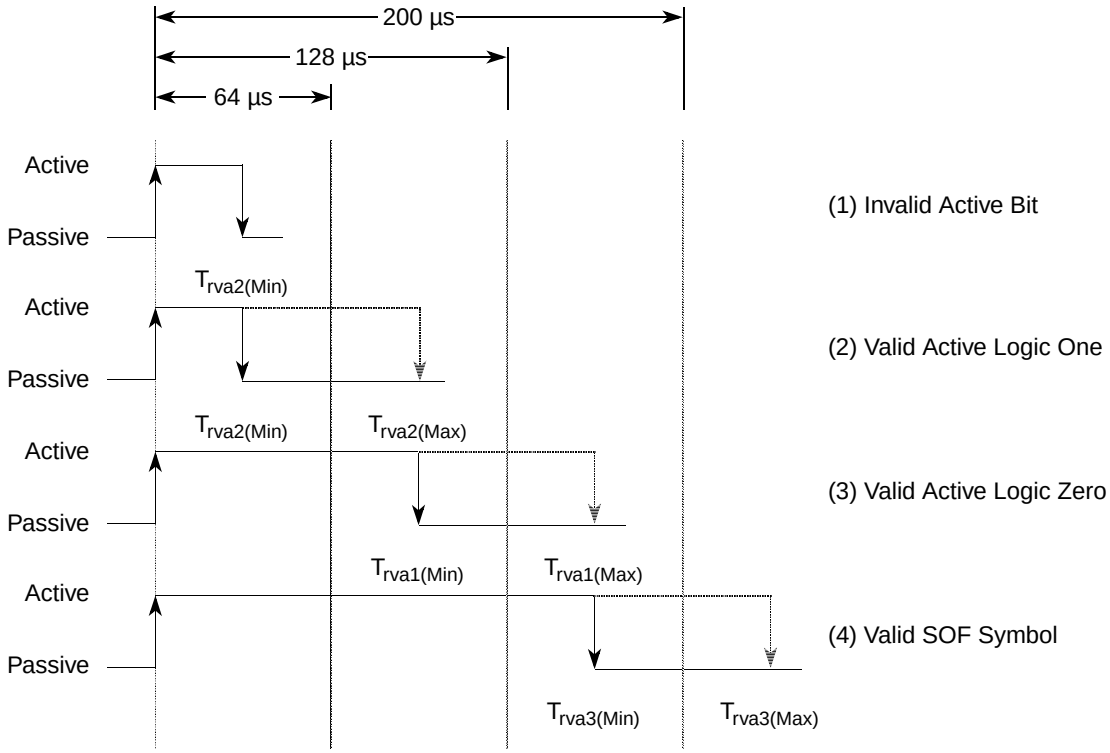


Figure 6-15. J1850 VPW Active Symbols

Invalid Active Bit

If the active to passive transition beginning the next data bit or symbol occurs between the passive to active transition beginning the current data bit or symbol and $T_{rva2(Min)}$, the current bit would be invalid. See Figure 6-15(1).

Valid Active Logic One

If the active to passive transition beginning the next data bit or symbol occurs between $T_{rva2(Min)}$ and $T_{rva2(Max)}$, the current bit would be considered a logic one. See Figure 6-15(2).

Valid Active Logic Zero

If the active to passive transition beginning the next data bit or symbol occurs between $T_{rva1(Min)}$ and $T_{rva1(Max)}$, the current bit would be considered a logic zero. See Figure 6-15(3).

Valid SOF Symbol

If the active to passive transition beginning the next data bit or symbol occurs between $T_{rva3(Min)}$ and $T_{rva3(Max)}$, the current symbol would be considered a valid SOF symbol. See Figure 6-15(4).

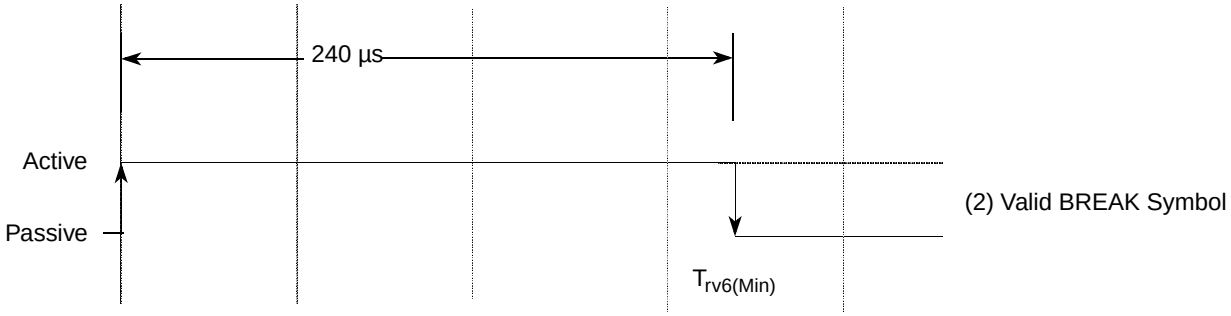


Figure 6-16. J1850 VPW BREAK Symbol

Valid BREAK Symbol

If the next active to passive transition does not occur until after $T_{rv6(Min)}$, the current symbol is considered a valid BREAK symbol. A BREAK symbol should be followed by a SOF symbol beginning the next message to be transmitted onto the J1850 bus. See Figure 6-16.

6.4.2.10.2 Message Arbitration

Message arbitration on the J1850 bus is accomplished in a non-destructive manner, allowing the message with the highest priority to be transmitted, while any transmitters that lose arbitration stop transmitting and wait for an idle bus to begin transmitting again.

If the BDLC module wishes to transmit onto the J1850 bus, but detects that another message is in progress, it automatically waits until the bus is idle. However, if multiple nodes begin to transmit in the same synchronization window, message arbitration occurs beginning with the first bit after the SOF symbol and continue with each bit thereafter.

The VPW symbols and J1850 bus electrical characteristics are carefully chosen so that a logic zero (active or passive type) always dominates over a logic one (active or passive type) simultaneously transmitted. Hence logic zeroes are said to be dominant and logic ones are said to be recessive.

When a node transmits a recessive bit and detects a dominant bit, it loses arbitration, and immediately stops transmitting. This is known as bitwise arbitration. The loss of arbitration flag in the BDLC_DLCBSVR register is set when arbitration is lost. If the interrupt enable bit (IE in the BDLC_DLCBCR1 register) is set, an interrupt request from the BDLC module is generated. Reading the BDLC_DLCBSVR register clears this flag.

During arbitration, or even throughout the transmitting message, when an opposite bit is detected, transmission is immediately stopped unless it occurs on the eighth bit of a byte. In this case, the BDLC module automatically appends as many as two extra 1 bits and then stops transmitting. These two extra bits are arbitrated normally and thus do not interfere with another message. The second 1 bit is not sent if the first loses arbitration. If the BDLC module has lost arbitration to another valid message, the two extra 1s do not corrupt the current message. However, if the BDLC module has lost arbitration due to noise on the bus, the two extra 1s ensure the current message is detected and ignored as a noise-corrupted message.

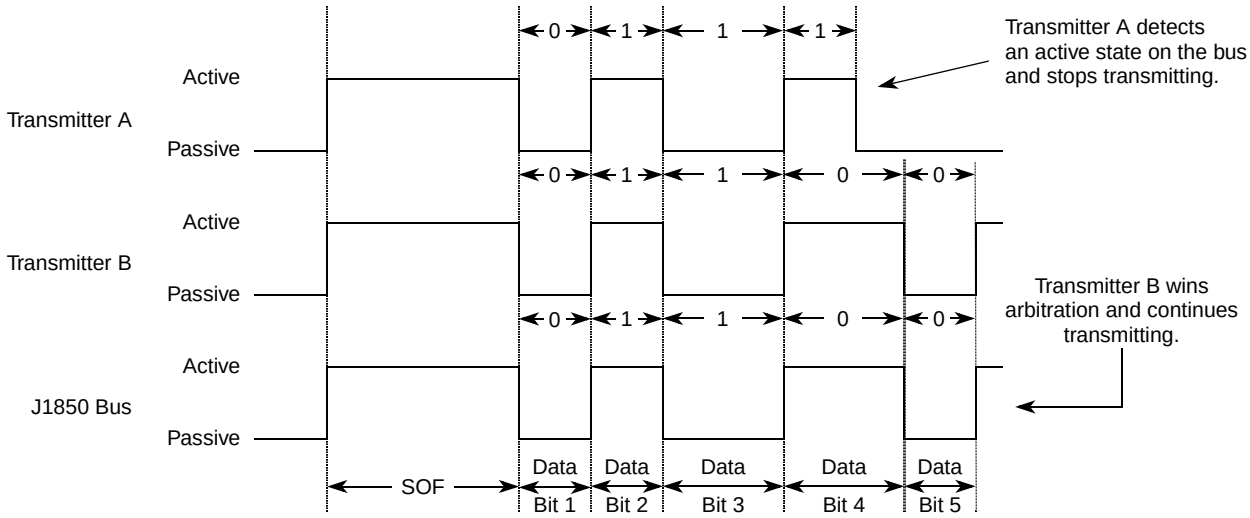


Figure 6-17. J1850 VPW Bitwise Arbitrations

Because a 0 dominates a 1, the message with the lowest value has the highest priority and always wins arbitration. A message with priority 000 wins arbitration over a message with priority 011. This method of arbitration works no matter how many bits of priority encoding are contained in the message.

6.4.2.11 J1850 Bus Errors

The BDLC module detects several types of transmit and receive errors that can occur during the transmission of a message onto the J1850 bus.

6.4.2.11.1 Transmission Error

If the BDLC module is transmitting a message and the message received contains a symbol error, a framing error, a bus fault, a BREAK symbol, or a logic 1 symbol when a logic 0 is being transmitted, this constitutes a transmission error. Receiving a logic 0 symbol when transmitting a logic 1 is considered a loss of arbitration condition (see Section 6.4.2.10.2, “Message Arbitration”) and not a transmission error. When a transmission error is detected, the BDLC module immediately ceases transmitting. Further transmission or reception is disabled until a valid EOF symbol is detected on the J1850 bus. The error condition is reflected by setting the symbol invalid or out of range flag in the BDLC_DLCBSVR register. If the interrupt enable bit (IE in the BDLC_DLCBCR1 register) is set, an interrupt request from the BDLC module is generated. Reading the BDLC_DLCBSVR register clears this flag.

6.4.2.11.2 CRC Error

A cyclical redundancy check (CRC) error is detected when the data bytes and CRC byte of a received message are processed, and the CRC calculation result is not equal to 0xC4. The CRC code should detect any single and 2 bit errors, as well as all 8 bit burst errors, and almost all other types of errors. The CRC error flag in the BDLC_DLCBSVR register is set when a CRC error is detected. If the interrupt enable bit (IE in the BDLC_DLCBCR1 register) is set, an interrupt request from the BDLC module is generated. Reading the BDLC_DLCBSVR register clears this flag.

6.4.2.11.3 Symbol Error

A symbol error is detected when an abnormal (invalid) symbol is detected in a message being received from the J1850 bus. See Invalid Passive Bit and Invalid Active Bit, which define invalid symbols. The symbol invalid or out of range flag in the BDLC_DLCBSVR register is set when a symbol error is detected. If the interrupt enable bit (IE in the BDLC_DLCBCR1 register) is set, an interrupt request from the BDLC module is generated. Reading the BDLC_DLCBSVR register clears this flag.

6.4.2.11.4 Framing Error

A framing error is detected when a received symbol occurs in an inappropriate location in the message frame. The following situations result in framing errors:

- An active logic 0 or logic 1 received as the first symbol of the frame.
- An SOF symbol received in any location other than the first symbol of a frame. Erroneous locations include: Within the data portion of a message or IFR; Immediately following the EOD in a message or IFR.
- An EOD symbol received on a non-byte boundary in a message or IFR.
- An active logic 0 or logic 1 received immediately following the EOD at the end of an IFR.

The symbol invalid or out of range flag in the BDLC_DLCBSVR register is set when a framing error is detected. If the interrupt enable bit (IE in the BDLC_DLCBCR1 register) is set, an interrupt request from the BDLC module is generated. Reading the BDLC_DLCBSVR register clears this flag.

6.4.2.11.5 Bus Fault

If a bus fault occurs, the response of the BDLC module depends upon the type of bus fault.

If the bus is shorted to V_{DD}, the BDLC module waits for the bus to fall to a passive state before it attempts to transmit a message. As long as the short remains, the BDLC never attempts to transmit a message onto the J1850 bus.

If the bus is shorted to ground, the BDLC module sees an idle bus, begin to transmit the message, and then detect a transmission error, since the short to ground would not allow the bus to be driven to the active (dominant) state. The BDLC module waits for assertion of the receive pin for (64 - analog round trip delay) t_{bdlc} cycles, after assertion of the transmit pin, before detecting the error. If the transmission is an IFR, the BDLC module waits for (280 - analog round trip delay) t_{bdlc} cycles before detecting an error. The analog round trip delay is determined by the value stored in the BDLC_DLCBARD register. The BDLC module sets the symbol invalid or out of range flag in the BDLC_DLCBSVR register, aborts that transmission, and waits for the next CPU command to transmit. In this case, the transmitter does not have to wait for an EOF symbol to be received to be enabled. If the interrupt enable bit (IE in the BDLC_DLCBCR1 register) is set, an interrupt request from the BDLC module is generated. Reading the BDLC_DLCBSVR register clears this flag.

If the bus fault is temporary, as soon as the fault is cleared, the BDLC module resumes normal operation. If the bus fault is permanent, it may result in permanent loss of communication on the J1850 bus.

6.4.2.11.6 Break

Any BDLC transmitting at the time a BREAK is detected treats the BREAK as if a transmission error had occurred, and halt transmission.

If while receiving a message the BDLC module detects a BREAK symbol, it treats the BREAK as a reception error.

If a BREAK symbol is received while the BDLC module is transmitting or receiving, the symbol invalid or out of range flag in the BDLC_DLCBSVR register is set. Further transmission/reception is disabled until the J1850 bus returns to the passive state and a valid EOF symbol is detected on the J1850 bus. If the interrupt enable bit (IE in the BDLC_DLCBCR1 register) is set, an interrupt request from the BDLC module is generated. Reading the BDLC_DLCBSVR register clears this flag.

The BDLC module can transmit a BREAK symbol. It can receive a BREAK symbol from the J1850 bus.

6.4.2.12 Bus Error Summary

The possible J1850 bus errors and the actions taken by the BDLC module are summarized in Table 6-22.

Table 6-22. BDLC Module J1850 Error Summary

Error Condition	BDLC Module Function
Transmission Error	BDLC module immediately ceases transmitting. Further transmission and reception is disabled until a valid EOF symbol is detected. The symbol invalid or out of range flag is set and interrupt generated if enabled.
Cyclical Redundancy Check (CRC) Error	CRC error flag set and interrupt generated if enabled.
Symbol Error	The symbol invalid or out of range flag is set and interrupt generated if enabled. Transmission and reception is disabled until a valid EOF symbol is detected.
Framing Error	The symbol invalid or out of range flag is set and interrupt generated if enabled. Transmission and reception is disabled until a valid EOF symbol is detected.
Bus short to V_{DD}.	The BDLC module does not transmit until short is corrected and a valid EOF is detected. Depending upon when short occurs and is corrected, this error condition may set the symbol invalid or out of range, CRC error, or loss of arbitration flags.
Bus short to GND.	Short is seen as an idle bus by BDLC module. If a transmission attempt is made before short is corrected, the symbol invalid or out of range flag is set and interrupt generated if enabled. Another transmission can be initiated as soon as short is corrected.
BREAK symbol reception	If doing so, the BDLC module immediately ceases transmitting. Symbol invalid or out of range flag set and interrupt generated if enabled.Transmission and reception is disabled until a valid EOF symbol is detected.

6.4.3 MUX Interface

The MUX Interface is responsible for bit encoding/decoding and digital noise filtering between the Protocol Handler and the Physical Interface. Refer to Figure 6-1.

6.4.3.1 MUX Interface—Rx Digital Filter

The Receiver section of the BDLC module includes a digital low pass filter to remove narrow noise pulses from the incoming message. An outline of the digital filter is shown in Figure 6-18.

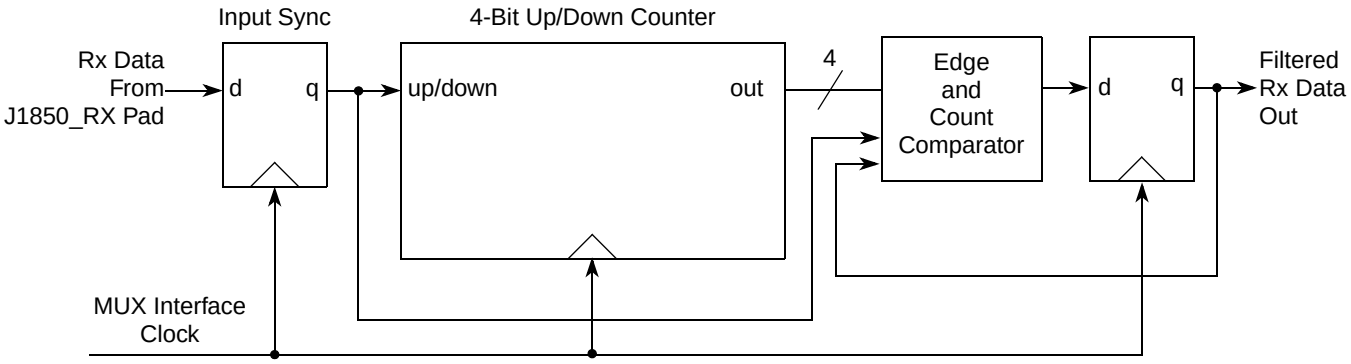


Figure 6-18. BDLC Module Rx Digital Filter Block Diagram

6.4.3.2 Operation

The clock for the digital filter is provided by the MUX Interface clock. At each positive edge of the clock signal, the current state of the Receiver input signal from the J1850_RX pad is sampled. The J1850_RX signal state is used to determine whether the counter should increment or decrement at the next positive edge of the clock signal.

The counter increments if the input data sample is high but decrement if the input sample is low. The counter then progresses up towards 15 if, on average, the J1850_RX signal remains high or progress down towards 0 if, on average, the J1850_RX signal remains low.

When the counter eventually reaches the value 15, the digital filter decides that the condition of the J1850_RX signal is at a stable logic level one and the Data Latch is set, causing the Filtered Rx Data signal to become a logic level one. Furthermore, the counter is prevented from overflowing and can only be decremented from this state.

Alternatively, should the counter eventually reach the value 0, the digital filter decides that the condition of the J1850_RX signal is at a stable logic level zero and the Data Latch is reset, causing the Filtered Rx Data signal to become a logic level zero. Furthermore, the counter is prevented from underflowing and can only be incremented from this state.

The Data Latch retains its value until the counter next reaches the opposite end point, signifying a definite transition of the J1850_RX signal.

6.4.3.3 Performance

The performance of the digital filter is best described in the time domain rather than the frequency domain.

If the signal on the J1850_RX signal transitions, there is a delay before that transition appears at the Filtered Rx Data output signal. This delay is between 15 and 16 clock periods, depending on where the

transition occurs with respect to the sampling points. This filter delay must be taken into account when performing message arbitration.

For example, if the frequency of the MUX Interface clock (f_{bdlc}) is 1.0486MHz, then the period (t_{bdlc}) is 954ns and the maximum filter delay in the absence of noise is 15.259 µs.

The effect of random noise on the J1850_RX signal depends on the characteristics of the noise itself. Narrow noise pulses on the J1850_RX signal is completely ignored if they are shorter than the filter delay. This provides a degree of low pass filtering.

If noise occurs during a symbol transition, the detection of that transition may be delayed by an amount equal to the length of the noise burst. This is a reflection of the uncertainty of where the transition is truly occurring within the noise.

Noise pulses that are wider than the filter delay, but narrower than the shortest allowable symbol length is detected by the next stage of the BDLC module's receiver as an invalid symbol.

Noise pulses that are longer than the shortest allowable symbol length is normally detected as an invalid symbol or as invalid data when the frame's CRC is checked.

6.4.4 Protocol Handler

The Protocol Handler is responsible for framing, collision detection, arbitration, CRC generation/checking, and error detection. The Protocol Handler conforms to SAE J1850 - Class B Data Communications Network Interface. Refer to Figure 6-1.

6.4.4.1 Protocol Architecture

The Protocol Handler contains the State Machine, Rx Shadow Register, Tx Shadow Register, Rx Shift Register, Tx Shift Register, and Loopback Multiplexer as shown in Figure 6-19.

6.4.4.1.1 Rx and Tx Shift Registers

The Rx Shift Register gathers received serial data bits from the J1850 bus and makes them available in parallel form to the Rx Shadow Register. The Tx Shift Register takes data, in parallel form, from the Tx Shadow Register and presents it serially to the State Machine so that it can be transmitted onto the J1850 bus.

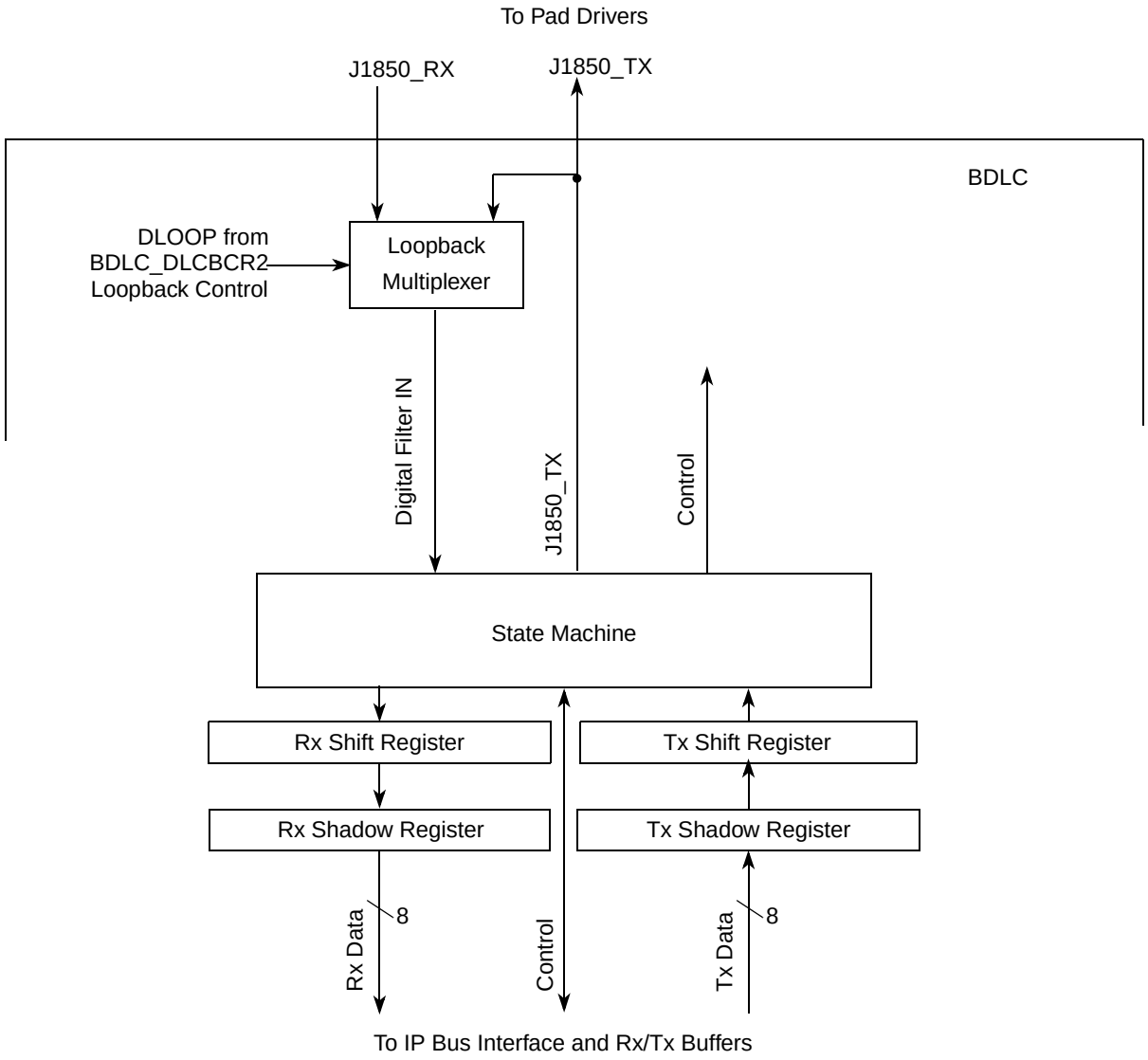


Figure 6-19. BDLC Protocol Handler Outline

6.4.4.1.2 Rx and Tx Shadow Registers

Immediately after the Rx Shift Register has completed shifting in a byte of data, this data is transferred to the Rx Shadow Register and RDRF or RXIFR is set and interrupt is generated if the interrupt enable bit (IE in the BDLC_DLCBCR1 register) is set. After the transfer takes place, this new data byte in the Rx Shadow Register is available to the CPU, and the Rx Shift Register is ready to shift in the next byte of data. Data in Rx Shadow Register must be retrieved by the CPU before it is overwritten by new data from the Rx Shift Register.

After the Tx Shift Register has completed its shifting operation for the current byte, the data byte in the Tx Shadow Register is loaded into the Tx Shift Register. After this transfer takes place, the Tx Shadow Register is ready to accept new data from the CPU.

6.4.4.1.3 Digital Loopback Multiplexer

The digital loopback multiplexer connects the input of the receive digital filter (See Figure 6-19) to either the transmit signal out to the pad (J1850_TX) or the receive signal from the pad (J1850_RX), depending on the DLOOP bit in BDLC Control Register 2 register.

6.4.4.1.4 State Machine

All of the functions associated with performing the protocol are executed or controlled by the State Machine. The State Machine is responsible for framing, collision detection, arbitration, CRC generation/checking, and error detection. The following sections describe the BDLC module's actions in a variety of situations.

6.4.4.1.5 4X Mode

The BDLC module can exist on the same J1850 bus as modules that use a special 4X (41.6 kbit/s) mode of J1850 VPW operation. The BDLC module can transmit and receive messages in 4X mode, if the 4XE bit is set in BDLC Control Register 2. If the 4XE bit is not set in the BDLC Control Register 2, any 4X message on the J1850 bus is treated as noise by the BDLC module and is ignored. Likewise, 4X messages transmitted on the SAE J1850 bus when the BDLC module is in normal mode is interpreted as noise on the network by the BDLC module.

6.4.4.1.6 Receiving a Message in Block Mode

Although not a part of the SAE J1850 protocol, the BDLC module allows for a special block mode of operation for the receiver. As far as the BDLC module is concerned, a block mode message is simply a long J1850 frame that contains an indefinite number of data bytes. All of the other features of the frame remain the same, including the SOF, CRC, and EOD symbols.

Another node wishing to send a block mode transmission must first inform all other nodes on the network that this is about to happen. This is usually accomplished by sending a special predefined message.

6.4.4.1.7 Transmitting a Message in Block Mode

A Block mode message is transmitted inherently by simply loading the bytes one by one into the BDLC Data Register register until the message is complete. The programmer should wait until the TDRE flag is set prior to writing a new byte of data into the BDLC Data Register register. The BDLC module does not contain any predefined maximum J1850 message length requirement.

6.4.5 Transmitting a Message

The design of the BDLC module enables the separate management of message reception and message transmission. All received messages can be managed almost identically, regardless of their origin.

This chapter only describes the steps necessary for transmitting a message and does not address the resulting reception of that message by the BDLC module. Message reception is described in Section 6.4.6, "Receiving A Message." Later sections deal with transmitting and receiving In-Frame Responses on the SAE J1850 bus.

6.4.5.1 BDLC Transmission Control Bits

Only one BDLC module control bit is used when transmitting a message onto the SAE J1850 bus. This bit, the Transmit End of Data (TEOD) bit, is set by the user to indicate to the BDLC module that the last byte of that part of the message frame has been loaded into the BDLC Data Register. The TEOD bit, located in BDLC Control Register 2, is also used when transmitting an In-Frame Response (IFR), but that usage is described in Section 6.4.7, “Transmitting an In-Frame Response (IFR).” Setting the TEOD bit indicates to the BDLC module that the last byte written to the BDLC Data Register is the final byte to be transmitted, and that following this byte a CRC byte and EOD symbol should be transmitted automatically. Setting the TEOD bit also inhibits any further TDRE interrupts until TEOD is cleared. The TEOD bit is cleared on the rising edge of the first bit of the transmitted CRC byte, or if an error or loss of arbitration is detected on the bus.

6.4.5.1.1 BDLC Data Register

The BDLC data register is a double-buffered register that handles the transmitted and received message bytes. Bytes to be transmitted onto the SAE J1850 bus are written to the BDLC data register, and bytes received from the bus by the BDLC module are read from the BDLC data register. Because this register is double buffered, bytes written into it cannot be read by the CPU. If this is attempted, the read byte is the last byte placed in the BDLC data register by the BDLC module, not the last byte written to the BDLC data register by the CPU. For an illustration of the BDLC data register, refer to Section 6.3.2.4, “BDLC Data Register (BDLC_DLCBDR).”

6.4.5.1.2 Transmitting a Message with the BDLC

To transmit a message using the BDLC module, the user writes the first byte of the message to be transmitted into the BDLC Data Register, initiating the transmission process. When the TDRE status appears in the BDLC_DLCBSVR register, the user writes the next byte into the BDLC Data Register. After all of the bytes have been loaded into the BDLC Data Register, the user sets the TEOD bit, and the BDLC module completes the message transmission. What follows is an overview of the basic steps required to transmit a message onto an SAE J1850 network using the BDLC module. For an illustration of this sequence, refer to Figure 6-20.

NOTE

Due to the byte-level architecture of the BDLC module, the 12-byte limit on message length as defined in SAE J1850 must be enforced by the user’s software. The number of bytes in a message (transmitted or received) has no meaning to the BDLC module.

1. Write the first byte into the BDLC data register.

   To initiate a message transmission, the CPU simply loads the first byte of the message to be transmitted into the BDLC Data Register. The BDLC module then performs the necessary bus acquisition duties to determine when the message transmission can begin.

   After the BDLC module determines that the SAE J1850 bus is free, a Start of Frame (SOF) symbol is transmitted, followed by the byte written to the BDLC Data Register. After the BDLC module readies this byte for transmission, the BDLC_DLCBSVR register reflects that the next byte can be written to the BDLC Data Register (TDRE interrupt).

NOTE

If the user writes the first byte of a message to be transmitted to the BDLC Data Register and then determines that a different message should be transmitted, the user can write a new byte to the BDLC Data Register up until the transmission begins. This new byte replaces the original byte in the BDLC Data Register.

2. When TDRE is indicated, write the next byte into the BDLC data register.

   When a TDRE state is reflected in the BDLC_DLCBSVR register, the CPU writes the next byte to be transmitted into the BDLC Data Register. This step is repeated until the last byte to be transmitted is written to the BDLC Data Register.

NOTE

Due to the design and operation of the BDLC module, when transmitting a message the user may write two, or possibly even three of the bytes to be transmitted into the BDLC Data Register before the first RDRF interrupt occurs. For this reason, the user should never use receive interrupts to control the sequencing of bytes to be transmitted.

3. Write the last byte to the BDLC data register and set TEOD.

   After the user has written the last byte to be transmitted into the BDLC Data Register, the user then sets the TEOD bit in BDLC Control Register 2. When the TEOD bit is set, once the byte written to the BDLC Data Register is transmitted onto the bus, the BDLC module begins transmitting the 8-bit CRC byte, as specified in SAE J1850. Following the CRC byte, the BDLC module transmits an EOD symbol onto the SAE J1850 bus, indicating that this part of the message has been completed. If no IFR bytes are transmitted following the EOD, an EOF is recognized and the message is complete.

   Setting the TEOD bit is the last step the CPU needs to take to complete the message transmission, and no further transmission-related interrupts occur. After the message has been completely received by the BDLC module, an EOF interrupt is generated. However, this is technically a receive function that can be managed by the message reception routine.

NOTE

While the TEOD bit is typically set immediately following the write of the last byte to the BDLC Data Register, it is also acceptable to wait until a TDRE interrupt is generated before setting the TEOD bit. While the example flowchart in Figure 6-20 shows the TEOD bit being set after the write to the BDLC Data Register, either method is correct. If a TDRE interrupt is pending, it is cleared when the TEOD bit is set.

6.4.5.2 Transmitting Exceptions

While this is the basic transmit flow, at times the message transmit process is interrupted. This can be due to a loss of arbitration to a higher priority message or due to an error being detected on the network. For the transmit routine, either of these events can be dealt with in a similar manner.

6.4.5.2.1 Loss of Arbitration

If a loss of arbitration (LOA) occurs while the BDLC module is transmitting onto the SAE J1850 bus, the BDLC module immediately stops transmitting, and a LOA status is reflected in the BDLC_DLCBSVR register. If the loss of arbitration has occurred on a byte boundary, an RDRF interrupt may also be pending once the LOA interrupt is cleared.

When a loss of arbitration occurs, the J1850 message handling software should immediately switch into the receive mode. If the TEOD bit was set, it is cleared automatically. If another attempt is to be made to transmit the same message, the user must start the transmit sequence over from the beginning of the message.

6.4.5.2.2 Error Detection

Similar to a loss of arbitration, if any error (except a CRC error) is detected on the SAE J1850 bus during a transmission, the BDLC module stops transmitting immediately. The transmitted byte is discarded, and the symbol invalid or out of range status is reflected in the BDLC_DLCBSVR register. As with the loss of arbitration, if the TEOD bit was set, it is cleared automatically and any attempt to transmit the same message has to start from the beginning.

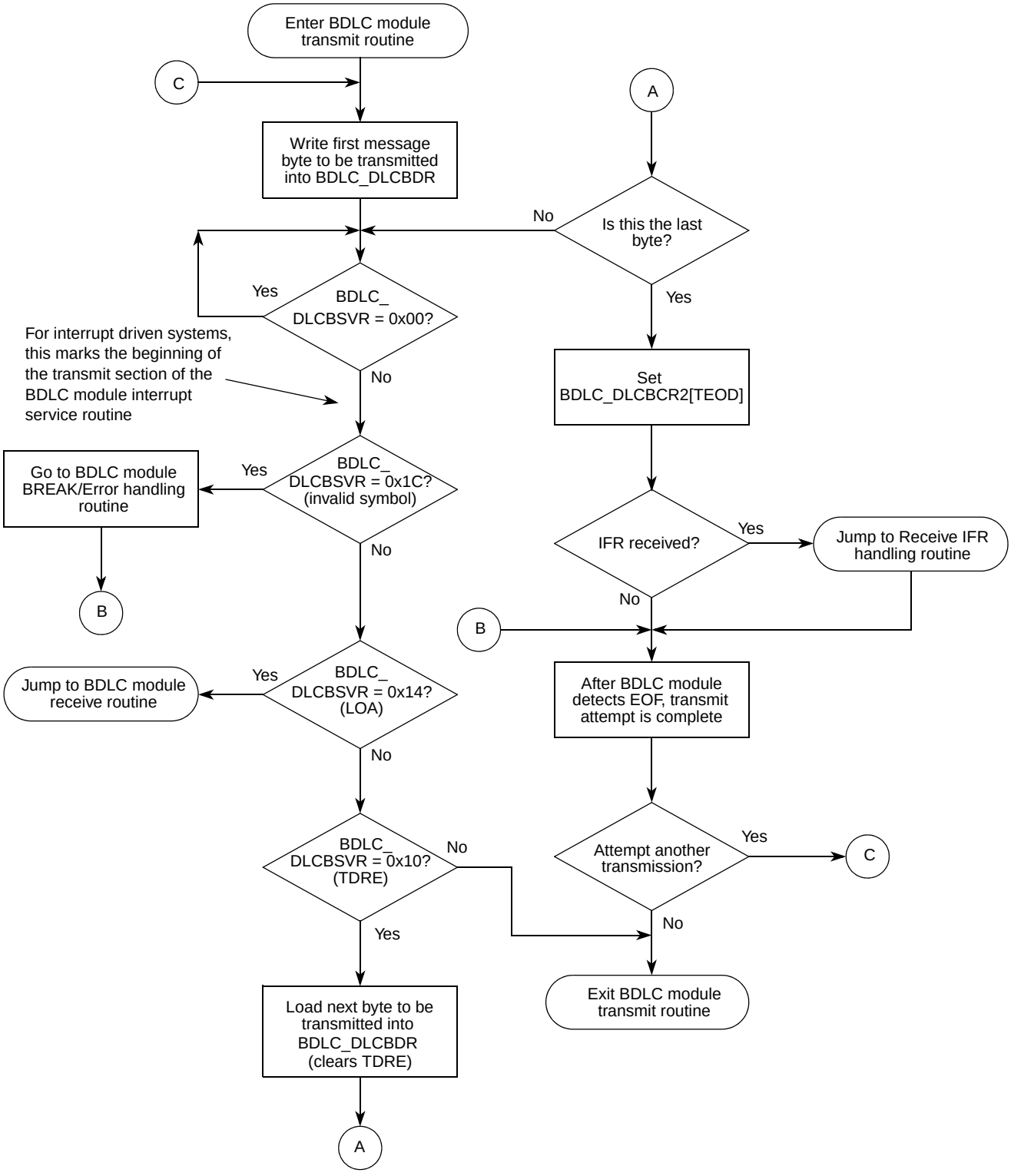
If a CRC error occurs following a transmission, this is also reflected in the BDLC_DLCBSVR register. However, since the CRC error is really a receive error based on the received CRC byte, at this point all bytes of the message have been transmitted. Thus, software must determine whether another attempt should be made to transmit the message in which the error occurred.

6.4.5.2.3 Transmitter Underrun

A transmitter underrun can occur when a TDRE interrupt is not serviced in a timely fashion. If the last byte loaded into the BDLC Data Register is completely transmitted onto the network before the next byte is loaded into the BDLC Data Register, a transmitter underrun occurs. If this does happen, the BDLC module transmits two additional logic ones to ensure that the partial message transmitted onto the bus does not end on a byte boundary. This is followed by an EOD and EOF symbol. The only indication to the CPU that an underrun occurred is the Symbol Invalid or Out of Range error indicated in the BDLC_DLCBSVR register. As with the other errors, software must determine whether another transmission attempt should be made.

6.4.5.2.4 In-Frame Response to a Transmitted Message

If an In-Frame Response (IFR) is received following the transmission of a message, the status indicating that an IFR byte has been received is indicated in the BDLC_DLCBSVR register before an EOF is indicated. Refer to Section 6.4.8, “Receiving An In-Frame Response (IFR),” for a description of how to manage the reception of IFR bytes.



NOTE: The EOF and CRC error interrupts are handled in the BDLC Module Receive Routine

Figure 6-20. Basic BDLC Transmit Flowchart

6.4.5.3 Aborting a Transmission

The BDLC module does not have a mechanism designed specifically for aborting a transmission. Because the module transmits each message on a byte-by-byte basis, there is little need to implement an abort mechanism. If the user has loaded a byte into the BDLC Data Register to initiate a message transmission and decides to send a different message, the byte in the BDLC Data Register can be replaced, right up to the point that the message transmission begins.

If the user has loaded a byte into the BDLC Data Register and then decides not to send any message at all, the user can let the byte transmit, and when the TDRE interrupt occurs let the transmitter underrun. This causes two extra logic ones followed by an EOF to be transmitted. While this method may require a small amount of bus bandwidth, the need to do this should be rare. Replacing the byte originally written to the BDLC Data Register with 0xFF also increases the probability of the transmitter losing arbitration if another node begins transmitting at the same time, also reducing the bus bandwidth needed.

6.4.6 Receiving A Message

The design of the BDLC module makes it especially easy to use for receiving messages off of the SAE J1850 bus. When the first byte of a message comes in, the BDLC_DLCBSVR register indicates to the CPU that a byte has been received. As each successive byte is received, it is in turn be reflected in the BDLC_DLCBSVR register. When the message is complete and the EOF has been detected on the bus, the BDLC_DLCBSVR register reflects this, indicating that the message is complete.

The basic steps required for receiving a message from the SAE J1850 bus are outlined below. For more information on receiving IFR bytes, refer to Section 6.4.8, “Receiving An In-Frame Response (IFR).”

6.4.6.1 BDLC Reception Control Bits

The only control bit used for message reception, the IMSG bit, is actually used to prevent message reception. When the IMSG bit is set, some of the BDLC module interrupts of the CPU are inhibited until the next SOF symbol is received. This allows the BDLC module to ignore the remainder of a message once the CPU has determined that it is of no interest. This helps reduce the amount of CPU overhead used to service messages received from the SAE J1850 network, since otherwise the BDLC module would require attention from the CPU for each byte broadcast on the network. The IMSG bit is cleared when the BDLC module receives an SOF symbol, or it can also be cleared by the CPU.

NOTE

While the IMSG bit can be used to prevent the CPU from having to service the BDLC module for every byte transmitted on the SAE J1850 bus, the IMSG bit should never be used to ignore the BDLC module’s own transmission. Because setting the IMSG bit prevents some BDLC_DLCBSVR register bits from being updated, it may be difficult for the CPU to complete the transmission correctly.

6.4.6.2 Receiving a Message with the BDLC Module

Receiving a message using the BDLC module is extremely straight-forward. As each byte of a message is received and placed into the BDLC Data Register, the BDLC module indicates this to the CPU with an Rx Data Register Full (RDRF) status in the BDLC_DLCBSVR register. When an EOF symbol is received, indicating to the CPU that the message is complete, this is reflected in the BDLC_DLCBSVR register.

Outlined below are the basic steps to be followed for receiving a message from the SAE J1850 bus with the BDLC module. For an illustration of this sequence, refer to Figure 6-21.

1. When an RDRF interrupt occurs, retrieve the data byte.

   When the first byte of a message following a valid SOF symbol is received that byte is placed in the BDLC Data Register, and an RDRF state is reflected in BDLC_DLCBSVR. No indication of the SOF reception is made, since the end of the previous message is marked by an EOF indication. The first RDRF state following this EOF indication should allow the user to determine when a new message begins.

   The RDRF interrupt is cleared when the received byte is read from the BDLC Data Register. After this is done, no further CPU intervention is necessary until the next byte is received, and this step is repeated.

   All bytes of the message, including the CRC byte, are placed into the BDLC Data Register as they are received for the CPU to retrieve.

2. When an EOF is received, the message is complete.

   After all bytes (including the CRC byte) have been received from the bus, the bus is idle for a time period equal to an EOD symbol. After the EOD symbol is received, the BDLC module verifies that the CRC byte is correct. If the CRC byte is not correct, this is reflected in BDLC_DLCBSVR.

   If no In-Frame Response bytes are transmitted following the EOD symbol, the EOD transitions into an EOF symbol. When the EOF is received it is reflected in BDLC_DLCBSVR, indicating to the user that the message is complete. If IFR bytes do follow the first EOD symbol, once they are complete another EOD is transmitted, followed by an EOF.

   After the EOF state is reflected in BDLC_DLCBSVR, this indicates to the user that the message is complete, and that when another byte is received it is the first byte of a new message.

6.4.6.3 Filtering Received Messages

No message filtering hardware is included on the BDLC module, so all message filtering functions must be performed in software. Because the BDLC module handles each message on a byte-by-byte basis, message filtering can be done as each byte is received, rather than after the entire message is complete. This enables the CPU to decide while a message remains in progress whether or not that message is of any interest.

At any point during a message, if the CPU determines that the message is of no interest the IMSG bit can be set. Setting the IMSG bit commands the BDLC module not to update the BDLC_DLCBSVR register until the next valid SOF is received. This prevents the CPU from having to service the BDLC module for each byte of every message sent over the network.

6.4.6.4 Receiving Exceptions

As with a message transmission, this basic message reception flow can be interrupted if errors are detected by the BDLC module. This can occur if an incorrect CRC is detected or if an invalid or out of range symbol appears on the SAE J1850 bus. A problem can also arise if the CPU fails to service the BDLC Data Register in a timely manner during a message reception.

6.4.6.4.1 Receiver Overrun

After a message byte has been received, the CPU must service the BDLC Data Register before the next byte is received, or the first byte is lost. If the BDLC Data Register is not serviced quickly enough, the next byte received is written over the previous byte in the BDLC Data Register. No receiver overrun indication is made to the CPU. If the CPU fails to service the BDLC module during the reception of an entire message, the byte remaining in the BDLC Data Register is the last byte received (usually a CRC byte).

After a receiver overrun occurs, there is no way for the CPU to recover the lost byte(s), so the entire message should be discarded. To prevent receiver overrun, the user should ensure that a BDLC RDRF interrupt is serviced before the next byte can be received. When polling the BDLC_DLCBSVR register, the user should select a polling interval that provides timely monitoring of the BDLC module.

6.4.6.4.2 CRC Error

If a CRC error is detected during a message reception, this is reflected in the BDLC_DLCBSVR register once an EOD time is recognized by the BDLC module. Because all bytes of the message have been received when this error is detected, software must ensure that all the received message bytes are discarded.

6.4.6.4.3 Invalid or Out of Range Symbol

If an invalid or out of range symbol, a framing error or a BREAK symbol is detected on the SAE J1850 bus during the reception of a message, the BDLC module immediately stops receiving the message and discard any partially received byte. The symbol invalid or out of range status is immediately reflected in the BDLC_DLCBSVR register. Following this, the BDLC module waits until the bus has been idle for a time period equal to an EOF symbol before receiving another message. As with the CRC error, the user should discard any partially received message if this occurs.

6.4.6.4.4 In-Frame Response to a Received Message

As mentioned above, if one or more IFR bytes are received following the reception of a message, the status indicating the reception of the IFR byte(s) is indicated in the BDLC_DLCBSVR register before the EOF is indicated. Refer to Section 6.4.8, “Receiving An In-Frame Response (IFR),” for a description of how to deal with the reception of IFR bytes.

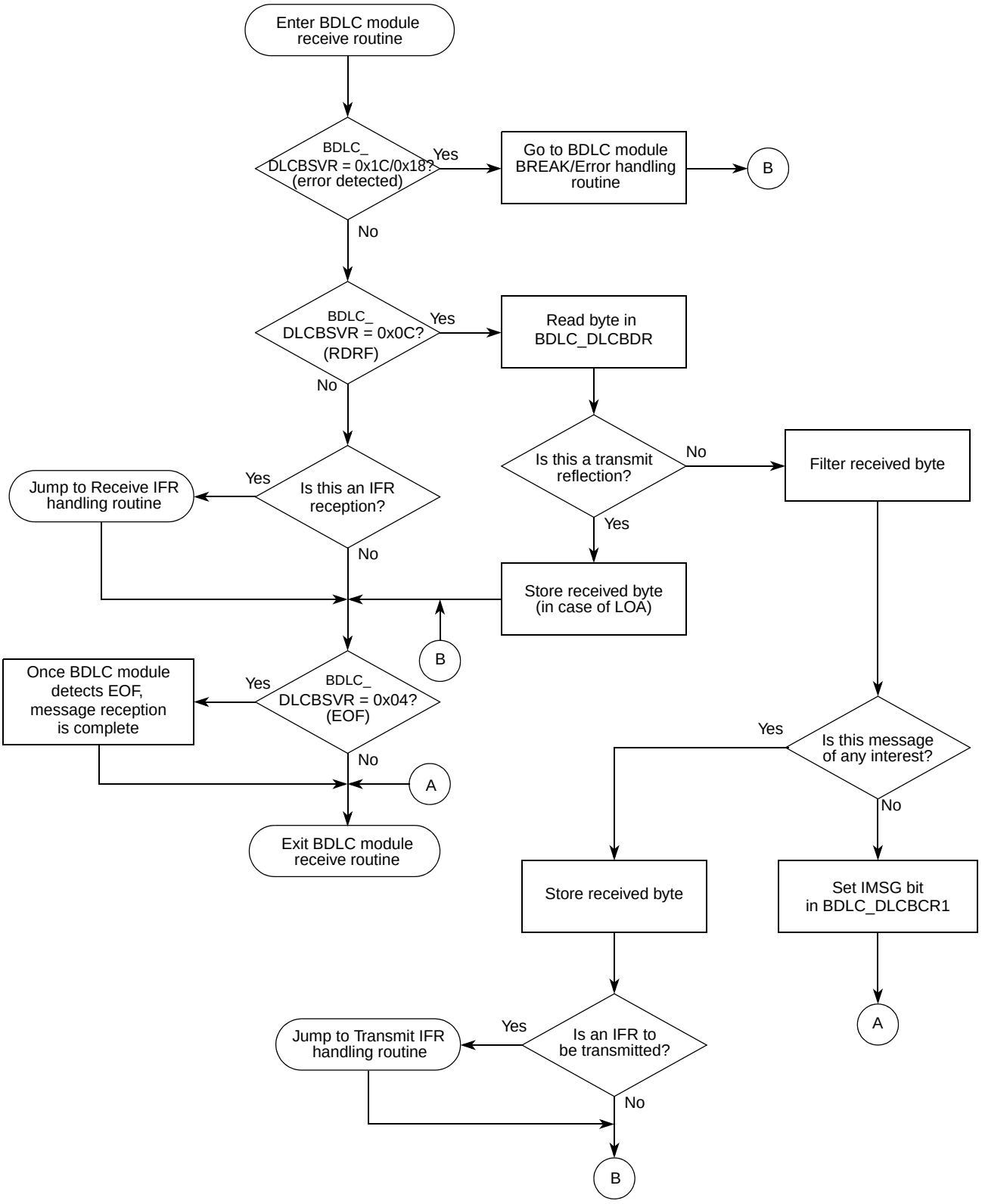


Figure 6-21. Basic BDLC Receive Flowchart

6.4.7 Transmitting an In-Frame Response (IFR)

The BDLC module can be used to transmit all four types of In-Frame Response (IFR) that are defined in SAE J1850. A brief definition of each IFR type is given below. For a more detailed description of each, refer the SAE J1850 document.

The explanation regarding IFR support by the BDLC module assumes familiarity with the use of IFRs as defined in SAE J1850 and understands the message header bit encoding and normalization bit formats used with the different types of IFRs. For more information on this, refer to the SAE J1850 document.

6.4.7.1 IFR Types Supported by the BDLC Module

SAE J1850 defines four distinct types of IFR. The first IFR is Type 0, or no IFR. IFR types 1, 2 and 3 are each made up of one or more bytes and, depending upon the type used, may be followed by a CRC byte. The BDLC module is designed to allow the user to transmit and receive all types of SAE J1850 IFRs, but only the network framing/error checking/bus acquisition duties are performed by the BDLC module. The user is responsible for determining the type of IFR to be transmitted, the number of retries to be made (if allowed), and the maximum number of bytes to be transmitted.

6.4.7.1.1 IFR Type 0: No Response

Generally, no IFR is used. The Type 0 IFR, as defined in SAE J1850, is no response. The EOD and EOF symbols follow directly after the CRC byte at the end of the message frame being transmitted. This type of IFR is inherently supported by the BDLC module, with no additional user intervention required.

6.4.7.1.2 IFR Type 1: Single Byte from a Single Responder

SAE J1850 defines the Type 1 IFR as a single byte from a single receiver. This type of IFR is used to acknowledge to the transmitter that the message frame was transmitted successfully on the network, and that at least one receiver received it correctly. A Type 1 IFR generally consists of the physical node ID of the receiver responding to the message, with no CRC byte appended. This type of response is used for broadcast-type messages, where there may be several intended receivers for a message, but the transmitter only wants to know that at least one node received it. In this case, all receivers begin transmitting their node ID following the EOD. Because all nodes on an SAE J1850 network have a unique node ID, if multiple nodes begin transmitting their node ID simultaneously, arbitration takes place. The node with the highest priority (lowest value) ID wins this arbitration process, and that node's ID makes up the IFR. No retries are attempted by the nodes that lose arbitration during a Type 1 IFR transmission.

A Type 1 IFR can also be used as a response to a physically addressed message, where the only intended receiver is the one that responds. In this case, no arbitration would take place during the IFR transmission, but the resulting IFR would consist of a single byte.

6.4.7.1.3 IFR Type 2: Single Byte from Multiple Responders

The Type 2 IFR, as defined in SAE J1850, is a series of single bytes, each transmitted by a different responder. This IFR type not only acknowledges to the transmitter that the message was transmitted successfully, but also reveals which receivers actually received the message. As with the Type 1 IFR, no CRC byte is appended to the end of a Type 2 IFR.

This IFR type is typically used with Function-type messages, where the original transmitter may need to know which nodes actually received the message. The basic difference between this type of IFR and the Type 1 IFR is that the nodes that lose arbitration while attempting to transmit their node ID during a Type 2 IFR wait until the byte that wins arbitration is transmitted and then again attempt to transmit their node ID onto the bus. The result is a series of node IDs, one from each receiver of the original message.

6.4.7.1.4 IFR Type 3: Multiple Bytes from a Single Responder

The last type of IFR defined by SAE J1850 is the Type 3 IFR. This IFR type consists of one or more bytes from a single responder. This type of IFR is used to return data to the original transmitter within the original message frame. This type of IFR may or may not have a CRC byte appended to it.

The Type 3 IFR is typically used with Function Read-type or Function Query-type messages, where the original transmitter is requesting data from the intended receiver. The node requesting the data transmits the initial portion of the message, and the intended receiver responds by transmitting the desired data in an IFR. In most cases, the original message requiring a Type 3 IFR is addressed to one particular node, so no arbitration should take place during the IFR portion of the message.

6.4.7.2 BDLC IFR Transmit Control Bits

The BDLC module has three bits that are used to control the transmission of an In-Frame Response. These bits, all located in BDLC Control Register 2, are TSIFR, TMIFR1, and TMIFR0. Each is used in conjunction with the TEOD bit to transmit one of three IFR types defined in SAE J1850. What follows is a brief description of each bit.

Because each of the bits used for transmitting an IFR with the BDLC module is used to transmit a particular type of IFR, only one bit should be set by the CPU at a time. However, should more than one of these bits get set at one time, a priority encoding scheme is used to determine which type of IFR is sent. This scheme prevents unpredictable operation caused by conflicting signals to the BDLC module. Table 6-23 illustrates which IFR bit is acted upon by the BDLC module should multiple IFR bits get set at the same time.

NOTE

As with transmitted messages, IFRs transmitted by the BDLC module are also received by the BDLC module. For a description of how IFR bytes received by the BDLC module should be handled, refer to Section 6.4.8, “Receiving An In-Frame Response (IFR).”

Table 6-23. IFR Control Bit Priority Encoding

READ/WRITE			ACTUAL		
TSIFR	TMIFR1	TMIFR0	TSIFR	TMIFR1	TMIFR0
0	0	0	0	0	0
1	—	—	1	0	0
0	1	—	0	1	0
0	0	1	0	0	1

6.4.7.3 Transmit Single Byte IFR

The Transmit Single Byte IFR (TSIFR) bit in BDLC Control Register 2 is used to transmit Type 1 and Type 2 IFRs onto the SAE J1850 bus. If this bit is set after a byte is loaded into the BDLC Data Register, the BDLC module attempts to send that byte, preceded by the appropriate Normalization Bit, as a single byte IFR without a CRC. If arbitration is lost, the BDLC module automatically attempts to transmit the byte again (without a Normalization Bit) as soon as the byte winning arbitration completes transmission. Attempts to transmit the byte continue until the byte is successfully transmitted, the TEOD bit is set by the user, or an error is detected on the bus.

The user must set the TSIFR bit before the EOD following the main part of the message frame is received, or no IFR transmit attempts are made for the current message. If another node does transmit an IFR to this message or a reception error occurs, the TSIFR bit is cleared. If not, the IFR is transmitted after the EOD of the next received message.

The TSIFR bit is automatically cleared after the EOD following one or more IFR bytes has been received or an error is detected on the bus.

6.4.7.4 Transmit Multi-Byte IFR 1

The Transmit Multi-Byte IFR 1 (TMIFR1) bit is used to transmit an SAE J1850 Type 3 IFR with a CRC byte appended. If this bit is set after the user has loaded the first byte of a multi-byte IFR into the BDLC Data Register, the BDLC module begins transmitting that byte, preceded by the appropriate Normalization Bit, onto the SAE J1850 bus. After this happens, a TDRE interrupt occurs, indicating to the user that the next IFR byte should be loaded into the BDLC Data Register. When the last byte to be transmitted is written to the BDLC Data Register, the user sets the TEOD bit. This causes a CRC byte and an EOD symbol to be transmitted following the last IFR byte.

As with the TSIFR bit, the TMIFR1 bit must be set before the EOD symbol is received or it remains cleared and no IFR transmit attempts are made. The TMIFR1 bit is cleared after the CRC byte and EOD are transmitted, if an error is detected on the bus, if a loss of arbitration occurs during the IFR transmission or if a transmitter underrun occurs when the user fails to service the TDRE interrupt in a timely manner. If a loss of arbitration occurs while the Type 3 IFR is being transmitted, transmission halts immediately and the loss of arbitration is indicated in the BDLC_DLCBSVR register.

6.4.7.5 Transmit Multi-Byte IFR 0

The Transmit Multi-Byte IFR 0 (TMIFR0) bit is used to transmit an SAE J1850 Type 3 IFR without a CRC byte appended. If this bit is set after the user has loaded the first byte of a multi-byte IFR into the BDLC Data Register, the BDLC module begins transmitting that byte, preceded by the appropriate Normalization Bit, onto the SAE J1850 bus. After this happens, a TDRE interrupt occurs, indicating to the user that the next IFR byte should be loaded into the BDLC Data Register. When the last byte to be transmitted is written to the BDLC Data Register, the user sets the TEOD bit. This causes an EOD symbol to be transmitted following the last IFR byte.

As with the TSIFR and TMIFR1 bits, the TMIFR0 bit must be set before the EOD symbol is received or it remains cleared and no IFR transmit attempts are made. The TMIFR0 bit is cleared after the CRC byte and EOD are transmitted, if an error is detected on the bus, if a loss of arbitration occurs during the IFR

transmission or if a transmitter underrun occurs when the user fails to service the TDRE interrupt in a timely manner. If a loss of arbitration occurs while the Type 3 IFR is being transmitted, transmission halts immediately and the loss of arbitration is indicated in the BDLC_DLCBSVR register.

NOTE

The TMIFR0 bit should not be used to transmit a Type 1 IFR. If a loss of arbitration occurs on the last bit of a byte being transmitted using the TMIFR0 bit, two extra logic ones are transmitted to ensure that the IFR does not end on a byte boundary. This can cause an error in a Type 1 IFR.

6.4.7.6 Transmitting An IFR with the BDLC module

While the design of the BDLC module makes the transmission of each type of IFR similar, the steps necessary for sending each is discussed. Again, a discussion of the bytes making up any particular IFR is not within the scope of this document. For a more detailed description of the use of IFRs on an SAE J1850 network, refer to the SAE J1850 document.

6.4.7.6.1 Transmitting a Type 1 IFR

To transmit a Type 1 IFR, the user loads the byte to be transmitted into the BDLC Data Register and sets both the TSIFR bit and the TEOD bit. This directs the BDLC module to attempt transmitting the byte written to the BDLC Data Register one time, preceded by the appropriate Normalization Bit. If the transmission is not successful, the byte is discarded and no further transmission attempts are made. For an illustration of the steps described below, refer to Figure 6-22.

1. Load the IFR byte into the BDLC data register.

   The user begins initiation of a Type 1 IFR by loading the desired IFR byte into the BDLC Data Register. If a byte has already been written into the BDLC Data Register for transmission as a new message, the user can simply write the IFR byte to the BDLC Data Register, replacing the previously written byte. This must be done before the first EOD symbol is received.

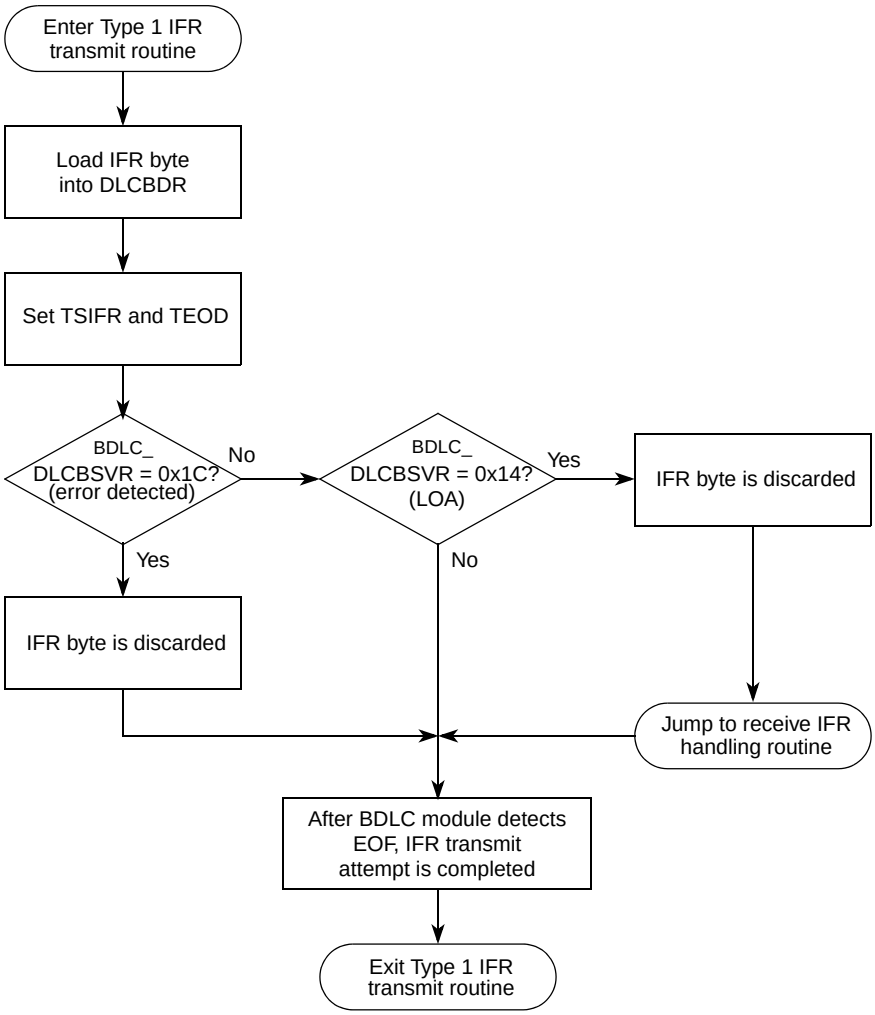


Figure 6-22. Transmitting A Type 1 IFR

2. Set the TSIFR and TEOD bits.

   The final step in transmitting a Type 1 IFR with the BDLC module is to set the TSIFR and TEOD bits in BDLC Control Register 2. Setting both bits directs the BDLC module to make one attempt at transmitting the byte in the BDLC Data Register as an IFR. If the byte is transmitted successfully, or if an error or loss of arbitration occurs, TEOD and TSIFR are cleared and no further transmit attempts are made.

6.4.7.6.2 Transmitting a Type 2 IFR

To transmit a Type 2 IFR, the user loads the byte to be transmitted into the BDLC Data Register and sets the TSIFR bit. After this is done, the BDLC module attempts to transmit the byte in the BDLC Data Register as a single byte IFR, preceded by the appropriate Normalization Bit. If the first BDLC module loses arbitration on the first attempt, it makes repeated attempts to transmit this byte until it is successful,

an error occurs or the user sets the TEOD bit. For an illustration of the steps described below, refer to Figure 6-23.

1. Load the IFR byte into the BDLC data register.

   As with the Type 1 IFR, the user begins initiation of a Type 2 IFR by loading the desired IFR byte into the BDLC Data Register. If a byte has already been written into the BDLC Data Register for transmission as a new message, the user can simply write the IFR byte to the BDLC Data Register, replacing the previously written byte. This must be done before the first EOD symbol is received.

2. Set the TSIFR bit.

   The second step necessary for transmitting a Type 2 IFR is to set the TSIFR bit in BDLC Control Register 2. Setting this bit directs the BDLC module to attempt to transmit the byte in the BDLC Data Register as an IFR until it is successful. If the byte is transmitted successfully or if an error or loss of arbitration occurs, TSIFR is cleared and no further transmit attempts are made.

3. If necessary, set the TEOD bit.

   The third step in transmitting a Type 2 IFR is only necessary if the user wishes to halt the transmission attempts. This may be necessary if the BDLC module's attempt to transmit the byte loaded into the BDLC Data Register continually loses arbitration, and the overall message length approaches the 12-byte limit as defined in SAE J1850.

   If it becomes necessary to halt the IFR transmission attempts, the user simply sets the TEOD bit in BDLC Control Register 2. If the BDLC module is between transmission attempts, it makes one more attempt to transmit the IFR byte. If it is transmitting the byte when TEOD is set, the BDLC module continues the transmission until it is successful or it loses arbitration to another transmitter. At this point, it then discards the byte and make no more transmit attempts.

NOTE

When transmitting a Type 2 IFR, the user should monitor the number of IFR bytes received to ensure that the overall message length does not exceed the 12-byte limit for the length of SAE J1850 messages. The user should set the TEOD bit when the 11th byte is received, which prevents the 12-byte limit from being exceeded.

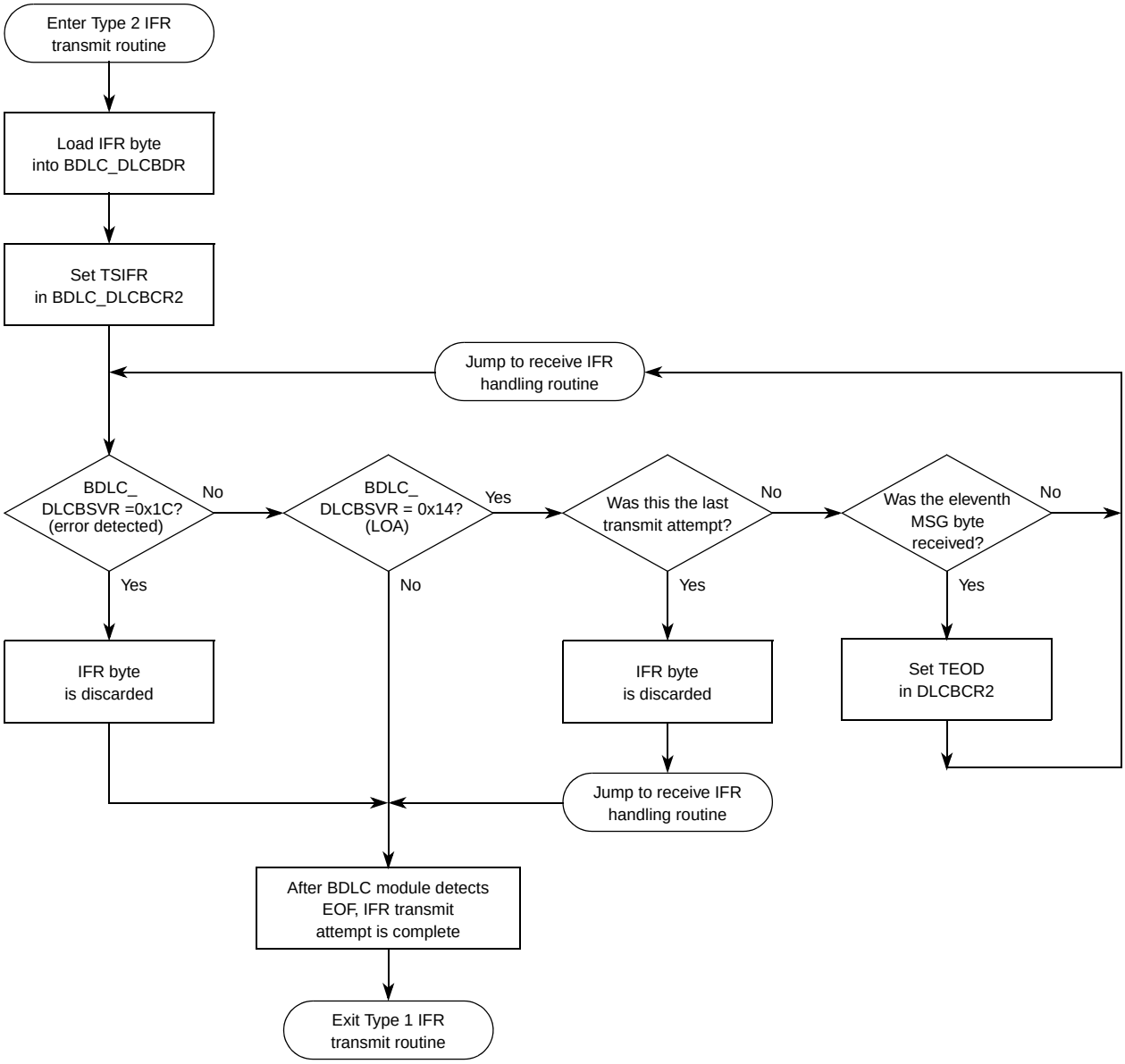
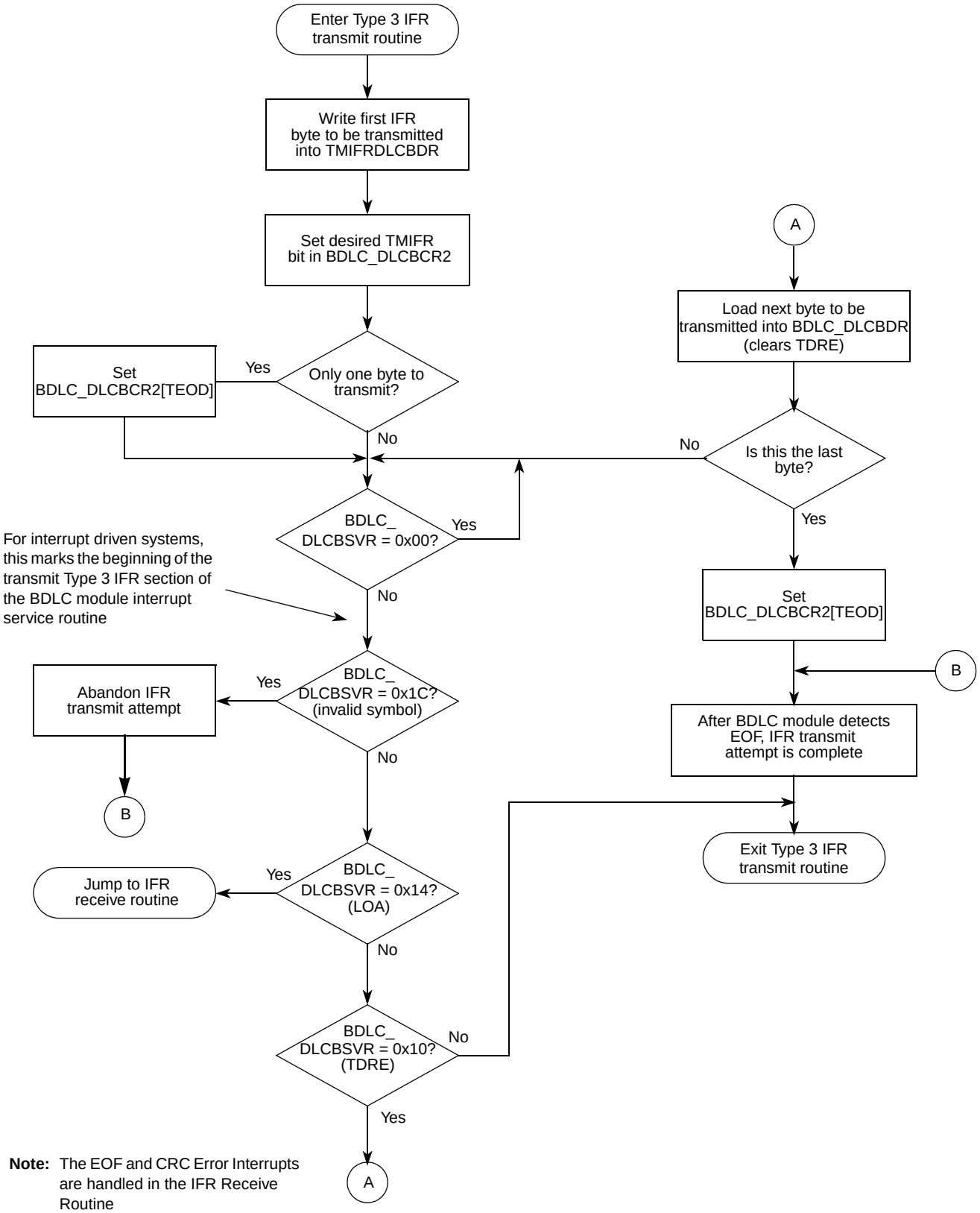


Figure 6-23. Transmitting A Type 2 IFR

6.4.7.6.3 Transmitting a Type 3 IFR

Transmitting a Type 3 IFR, with or without a CRC byte, is done in a fashion similar to transmitting a message frame. The user loads the first byte to be transmitted into the BDLC Data Register and then sets the appropriate TMIFR bit, depending upon whether a CRC byte is desired. When the last byte is written to the BDLC Data Register, the TEOD bit is set, and a CRC byte (if desired) and an EOD are then transmitted. Because the two versions of the Type 3 IFR are transmitted identically, the description that follows discusses both. For an illustration of the Type 3 IFR transmit sequence, refer to Figure 6-24.



Note: The EOF and CRC Error Interrupts are handled in the IFR Receive Routine

Figure 6-24. Transmitting A Type 3 IFR

1. Load the first IFR byte into the BDLC data register.

   The user begins initiation of a Type 3 IFR, as with each of the other IFR types, by loading the desired IFR byte into the BDLC Data Register. If a byte has already been written into the BDLC Data Register for transmission as a new message, the user can simply write the first IFR byte to the BDLC Data Register, replacing the previously written byte. This must be done before the first EOD symbol is received.

2. Set the TMIFR bit.

   The second step necessary for transmitting a Type 3 IFR is to set the desired TMIFR bit in BDLC Control Register 2, depending upon whether or not a CRC is desired. As previously described in Section 6.4.7.2, "BDLC IFR Transmit Control Bits," the TMIFR1 bit should be set if the user requires a CRC byte to be appended following the last byte of the Type 3 IFR, and TMIFR0 if no CRC byte is required.

   Setting the TMIFR1 or TMIFR0 bit directs the BDLC module to transmit the byte in the BDLC Data Register as the first byte of a single or multi-byte IFR preceded by the appropriate Normalization Bit. After this has occurred, the BDLC_DLCBSVR register reflects that the next byte of the IFR can be written to the BDLC Data Register (TDRE interrupt).

   NOTE

   The user must set the TMIFR1 or TMIFR0 bit before the EOD following the main part of the message frame is received or no IFR transmit attempts are made for the current message. If another node does transmit an IFR to this message or a reception error occurs, the TMIFR1 or TMIFR0 bit is cleared. If not, the IFR is transmitted after the EOD of the next received message.

3. When TDRE is indicated, write the next IFR byte into the BDLC data register.

   When a TDRE state is reflected in the BDLC_DLCBSVR register, the CPU writes the next IFR byte to be transmitted into the BDLC Data Register, clearing the TDRE interrupt. This step is repeated until the last IFR byte to be transmitted is written to the BDLC Data Register.

   NOTE

   When transmitting a Type 3 IFR, you may write two or three of the bytes to be transmitted into the BDLC Data Register before the first Rx IFR interrupt occurs. For this reason, never use receive IFR byte interrupts to control the sequencing of IFR bytes to be transmitted.

4. Write the last IFR byte into the BDLC data register and set TEOD.

   After the last IFR byte to be transmitted is written to the BDLC Data Register, the CPU then sets the TEOD bit in BDLC Control Register 2. After the TEOD bit is set and the last IFR byte written to the BDLC Data Register is transmitted onto the bus (if the TMIFR1 bit has been set), the BDLC module begins transmitting the CRC byte, followed by an EOD. If the TMIFR0 bit has been set, the last IFR byte is immediately followed by the transmission of an EOD. Following the EOD, EOF is recognized and the message is complete.

   If a loss of arbitration occurs at any time during the transmission of a Type 3 IFR, the TMIFR bit is set, and the TEOD bit (if set) is cleared, any IFR byte being transmitted is discarded and the loss of arbitration state is reflected in the BDLC_DLCBSVR register. Likewise, if an error is detected

during the transmission of a Type 3 IFR the IFR control bits are cleared, the byte being transmitted is discarded, and the BDLC_DLCBSVR register reflects the detected error.

If the Type 3 IFR being transmitted is made up of a single byte, the appropriate TMIFR bit and the TEOD bit can be set at the same time. The BDLC module then treats that byte as the first and last IFR byte to be sent.

6.4.7.7 Transmitting IFR Exceptions

This basic IFR transmitting flow can be interrupted for the same reasons as a normal message transmission. The IFR transmit process can be adversely affected due to a loss of arbitration, an Invalid or Out of Range Symbol, or due to a transmitter underrun caused by the CPU failing to service a TDRE interrupt in a timely fashion. For a description of how these exceptions can affect the IFR transmit process, refer to Section 6.4.5.2, “Transmitting Exceptions.”

6.4.8 Receiving An In-Frame Response (IFR)

Receiving an In-Frame Response with the BDLC module is similar to receiving a message frame. As each byte of an IFR is received, the BDLC_DLCBSVR register indicates this to the CPU. An EOF indication in the BDLC_DLCBSVR register indicates that the IFR (and message) is complete. Also, the IMSG bit can also be used to command the BDLC module to mask any further network activity from the CPU, including IFR bytes being received, until the next valid SOF is received.

NOTE

As with a message transmission, the IMSG bit should never be used to ignore the BDLC module’s own IFR transmissions. This is again due to the BDLC_DLCBSVR register bits being inhibited from updating until IMSG is cleared, preventing the CPU from detecting any IFR-related state changes that may be of interest.

6.4.8.1 Receiving an IFR with the BDLC Module

Receiving an IFR from the SAE J1850 bus requires the same procedure that receiving a message does, except that as each byte is received the Received IFR Byte (RxIFR) state is indicated in the BDLC_DLCBSVR register. All other actions are the same. For an illustration of the steps described below, refer to Figure 6-25.

1. When RxIFR Interrupt occurs, retrieve IFR byte.

   When the first byte of an IFR following a valid EOD symbol is received that byte is placed in the BDLC Data Register, and an RxIFR state is reflected in the BDLC_DLCBSVR register. No indication of the EOD reception in made because the RxIFR state indicates that the main portion of the message has ended and the IFR portion has begun.

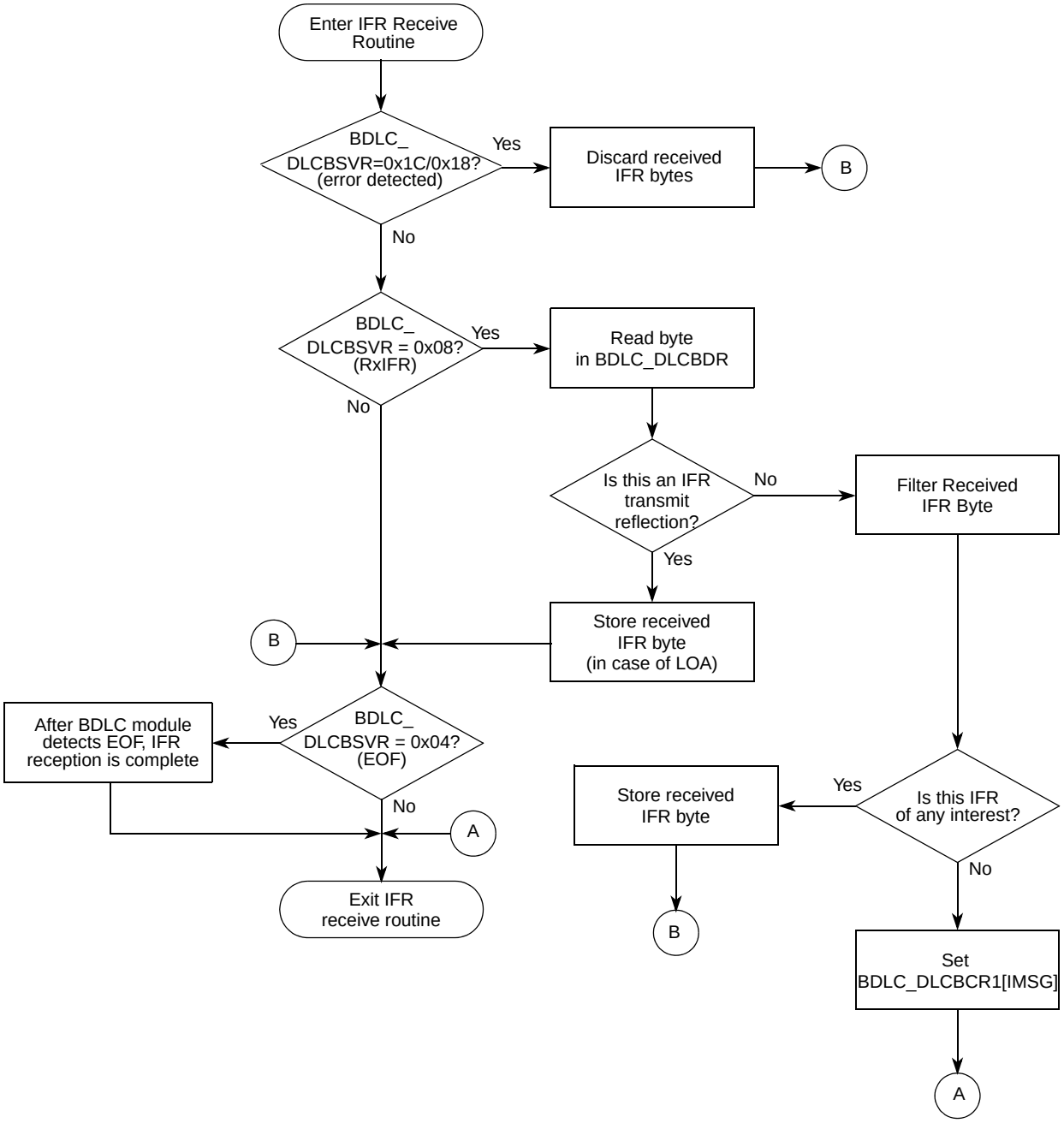


Figure 6-25. Receiving An IFR with the BDLC Module

The RxIFR interrupt is cleared when the received IFR byte is read from the BDLC Data Register. After this is done, no further CPU intervention is necessary until the next IFR byte is received, and this step is repeated. As with a message reception, all bytes of the IFR, including the CRC byte, are placed into the BDLC Data Register as they are received for the CPU to retrieve.

2. When an EOF is received, the IFR (and Message) is complete.

   After all IFR bytes (including the possible CRC byte) have been received from the bus, the bus is idle again for a time period equal to an EOD symbol. Following this, the BDLC module determines

whether or not the last byte of the IFR is a CRC byte, and if so verify that the CRC byte is correct. If the CRC byte is not correct, this is reflected in the BDLC_DLCBSVR register.

After an additional period of time, the EOD symbol transitions into an EOF symbol. When the EOF is received, it is reflected in BDLC_DLCBSVR, indicating to the user that the IFR and the message is complete.

6.4.8.2 Receiving IFR Exceptions

This basic IFR receiving flow can be interrupted for the same reasons as a normal message reception. The IFR receiving process can be adversely affected due to a CRC error, an Invalid or Out of Range Symbol or due to a receiver overrun caused by the CPU failing to service an RxIFR interrupt in a timely fashion. For a description of how these exceptions can affect the IFR receiving process, refer to Section 6.4.6.4, “Receiving Exceptions.”

6.4.9 Special BDLC Module Operations

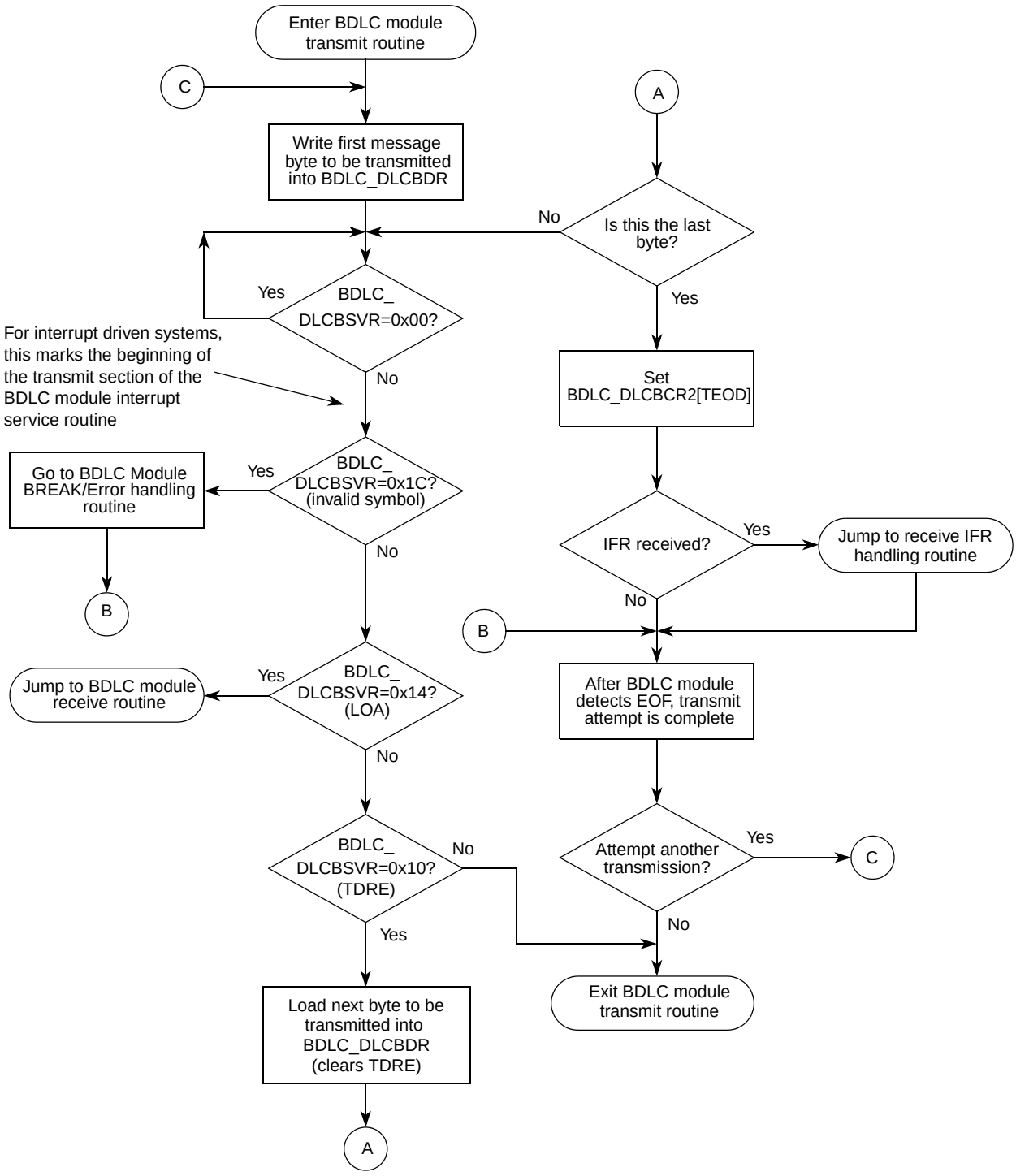
There are a few special operations that the BDLC module can perform. What follows is a brief description of each of these functions and when they might be used.

6.4.9.1 Transmitting Or Receiving A Block Mode Message

The BDLC module, because it handles each message on a byte-by-byte basis, has the inherent capability of handling messages any number of bytes in length. While during normal operation this requires the user to carefully monitor message lengths to ensure compliance with SAE J1850 message limits, often in a production or diagnostic environment messages that exceed the SAE J1850 limits can be beneficial. This is especially true when large amounts of configuration data need to be downloaded over the SAE J1850 network.

Because of the BDLC module’s architecture, it can both transmit and receive messages of unlimited length. The CRC calculations for transmitting and receiving are not limited to eight bytes, but are instead calculated and verified using all bytes in the message, regardless of the number. All control bits, including TEOD and IMSG, also work in an identical manner, regardless of the length of the message.

To transmit or receive these block mode messages, no extra BDLC module control functions must be performed. The user simply transmits or receives as many bytes as desired in one message frame, and the BDLC module operates as if a message of normal length was being used.



NOTE: The EOF and CRC Error Interrupts are handled in the BDLC Module Receive Routine

Figure 6-26. Basic BDLC Module Transmit Flowchart

6.4.9.2 Transmitting Or Receiving A Message In 4X Mode

In a diagnostic or production environment large amounts of data may need to be downloaded across the network to a component or module. This data is often sent in a large block mode message (see above) that violates the SAE J1850 limit for message length. To speed up the downloading of these large blocks of data, they are sometimes transmitted at four times (4X) the normal bit rate for the Variable Pulse Width modulation version of SAE J1850. This higher speed transmission, nominally 41.6 kbit/s, allows these large blocks to be transmitted much more quickly.

The BDLC module is designed to receive and transmit messages at this higher speed. By setting the 4XE bit in BDLC Control Register 2, the user can command the BDLC module to transmit and receive any message over the network at a 4X rate.

If the BDLC module is placed in this 4X mode, messages transmitted at the normal bit rate are not received correctly. Likewise, 4X messages transmitted on the SAE J1850 bus when the BDLC module is in normal mode are interpreted as noise on the network by the BDLC module. For more information on the 4XE bit, refer to Section 6.4.4.1.5, “4X Mode.”

6.5 Initialization Information

To initialize the BDLC module, the user should first write the desired data to the configuration bits. The BDLC module should then be taken out of digital and analog loopback mode and enabled. Exiting from loopback mode entails change of state indications in the BDLC_DLCBSVR register that must be dealt with. After this is complete, CPU interrupts can be enabled (if desired), and then the BDLC module is capable of SAE J1850 serial network communication. For an illustration of the sequence necessary for initializing the BDLC module, refer to Figure 6-27.

6.5.1 Initializing the Configuration Bits

The first step necessary for initializing the BDLC module following an MCU reset is to write the desired values to each of the BDLC module control registers. This is best done by storing predetermined initialization values directly into these registers. The following description outlines a basic flow for initializing the BDLC module. This basic flow does not detail more elaborate initialization routines, such as performing digital and analog loopback tests before enabling the BDLC module for SAE J1850 communication. However, from the following descriptions and the BDLC module specification, the user should be able to develop routines for performing various diagnostic procedures such as loopback tests.

1. Initialize the BDLC_DLCBARD register.

   Begin initialization of the configuration bits by writing the desired analog transceiver configuration data into the the BDLC_DLCBARD register. Following this write to the BDLC_DLCBARD register, all of these bits become read only.

2. Initialize BDLC Baud Rate Select Register.

   The next step in BDLC module initialization is to write the desired bus clock divisor minus one into the BDLC Baud Rate Select Register. The divisor should be chosen to generate a 1 MHz or 1.048576 MHz MUX interface clock (f_{bdlc}). Following this write to BDLC Baud Rate Select Register, all of these bits become read only.

3. Initialize BDLC Control Register 2.

The next step in BDLC module initialization should be writing the configuration bits into the BDLC Control Register 2 register. This initialization description assumes the BDLC module is put into normal mode (not 4X mode), and that the BDLC module should not yet exit digital or analog loopback mode. Therefore, this step should write SMRST and DLOOP as logic ones, 4XE as a logic zero, write NBFS to the desired level, and write TEOD, TSIFR, TMIFR1, and TMIFR0 as logic zeros. These last four bits MUST be written as logic zeros to prevent undesired operation of the BDLC module.

4. Initialize the BDLC_DLCBCR1 register.

   The next step in BDLC module initialization is to write the configuration bits in the BDLC_DLCBCR1 register. The CLKS bit should be written to its desired values at this time, following which it becomes read-only. The IE bit should be written as a logic zero at this time so BDLC module interrupts of the CPU remain masked for the time being. The IMSG bit should be written as a logic one to prevent any receive events from setting the BDLC_DLCBSVR register until a valid SOF (or BREAK) symbol has been received by the BDLC module.

6.5.2 Exiting Loopback Mode and Enabling the BDLC Module

After the configuration bits have been written to the desired values, the BDLC module should be taken out of loopback and connected to the SAE J1850 bus. This is done by clearing the DLOOP bit and then setting the BDLCE bit in the BDLC Control Register.

1. Perform Loopback Tests (optional)

   After the BDLC module is configured for desired operation, the user may wish to perform digital and/or analog loopback tests to determine the integrity of the link to the SAE J1850 network. This would involve leaving the DLOOP bit (BDLC Control Register 2) set, setting the BDLCE bit, preforming the desired loopback tests and finally exiting digital loopback mode by clearing DLOOP in the BDLC Control Register 2.

2. Exit Loopback Mode and Enable the BDLC Module

   If loopback mode tests are not to be preformed the BDLC module can be removed from digital loopback mode by clearing the DLOOP bit. The BDLC module can then be enabled by setting the BDLCE bit in the BDLC Control Register.

After DLOOP is cleared and BDLCE is set, the BDLC module is ready for SAE J1850 communication. However, to ensure that the BDLC module does not attempt to receive a message already in progress or to transmit a message while another device is transmitting, the BDLC module must first observe an EOF symbol on the bus before the receiver is activated. To activate the transmitter, the BDLC module needs to observe an Inter-Frame Separator symbol.

6.5.3 Enabling BDLC Interrupts

The final step in readying the BDLC module for proper communication is to clear any pending interrupt sources and then, if desired, enable BDLC module interrupts of the CPU.

1. Clear Pending BDLC Interrupts

   To ensure that the BDLC module does not immediately generate a CPU interrupt when interrupts are enabled, the user should read the BDLC_DLCBSVR register to determine if any BDLC module

interrupt sources are pending before setting the IE bit in the BDLC Control Register 1. If BDLC_DLCBSVR = 0, no interrupts are pending and the user is free to enable BDLC interrupts, if desired.

If BDLC_DLCBSVR indicates that an interrupt is pending, the user should perform whatever actions are necessary to clear the interrupt source before enabling the interrupts. Whether any interrupts are pending depends primarily upon how much time passes between the exit from loopback modes and enabling the BDLC module and the enabling of interrupts. It is a good practice to always clear any source of interrupts before enabling interrupts on any MCU subsystem.

If any interrupts are pending (BDLC_DLCBSVR not 0), then each interrupt source should be dealt with accordingly. After all of the interrupt sources have been dealt with, BDLC_DLCBSVR should read 0, and the user is then free to enable BDLC interrupts.

2. Enable BDLC Interrupts

   The last step in initializing the BDLC module is to enable interrupts to the CPU, if so desired. This is done by simply setting the IE bit in the BDLC_DLCBCR1 register. Following this, the BDLC module is ready for operating in interrupt mode. If the user chooses not to enable interrupts, BDLC_DLCBSVR must be polled periodically to ensure that state changes in the BDLC module are detected and dealt with appropriately.

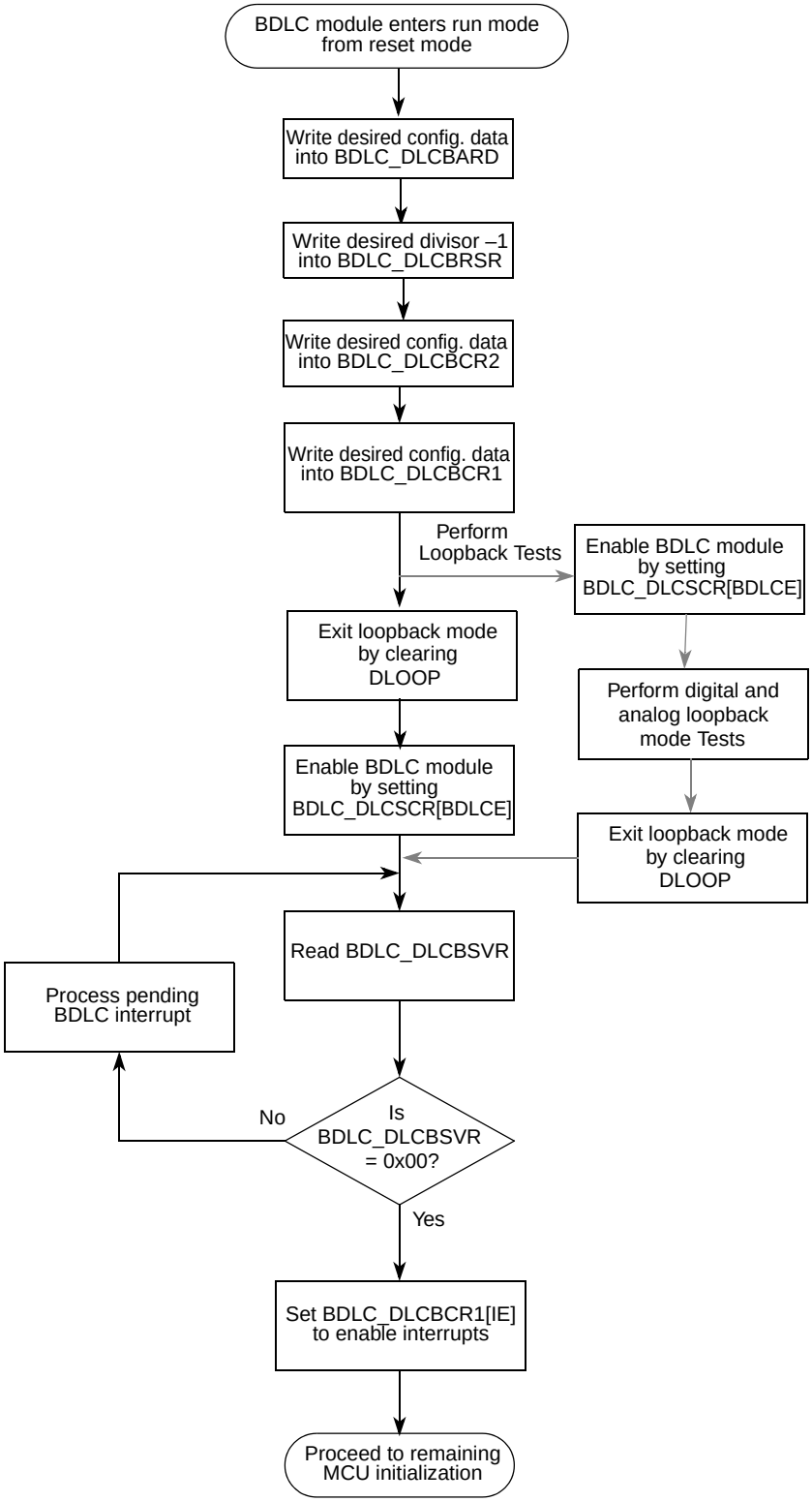


Figure 6-27. Basic BDLC Module initialization Flowchart

Chapter 7
CPU e300 Core Power Architecture

7.1 Introduction

The following sections are contained in this chapter:

- e300c4 Processor Core Functional Overview
- e300c4 Core Reference Manual
- Unsupported e300c4 Core Features

7.2 e300c4 Processor Core Functional Overview

The module integrates a e300c4 processor core, which is based on and compatible with the 603e Power Architecture-compliant microprocessor. The e300c4 core is completely embedded, as its address, data, and control signals are not externally visible. The e300c4 core has the following features:

- e300 Power Architecture Core
- Dual Issue, superscalar architecture
- 32 KB instruction cache, 32 KB data cache
- Double precision FPU
- Instruction and data MMU
- Power management modes:
  - Nap
  - Doze
  - Sleep
- Standard and critical interrupt capability

For additional information on the capabilities and features of the e300c4 core, refer to e300 user documentation.

After power-on or hard reset, initial boot instructions are fetched from the LocalPlus bus, with CS0 active, or from page 0 of NAND flash. To facilitate high speed execution, boot code is typically copied from a flash or ROM device to SDRAM. The e300c4 core can execute code from the on-chip SRAM.

The e300c4 core has memory mapped access to all resources, including:

- All on-chip programming registers
- External SDRAM
- Internal SRAM

Bursting is supported on the Coherent Systems Bus (CSB). Critical word first protocol is employed when the e300c4 core attempts to fill its address and data caches. The ID values for the MPC5125 are shown in Table 7-1.

Table 7-1. ID Values for the MPC5125

	M01S Mask Set
PVR	0x8086_2010
MPC5125 SVR	0x8019_0010
JTAG ID Code	0x0540_C01D

7.3 e300c4 Core Reference Manual

A complete specification for the e300c4 core implementation used on the module is obtained through a collection of documentation.

- Power Architecture Microprocessor Family: The Programming Environments for 32-bit Microprocessors, Rev. 2: MPCFPE32B/AD
- e300 Power Architecture Core Family Reference Manual, Rev. 4

The programming environments manual provides information about resources defined by the Power Architecture architecture common to Power Architecture processors. Implementation variances relative to Rev. 4 of the Programming Environments Manual are available in the e300 Core Reference Manual.

The e300 Power Architecture Core Family Reference Manual can be obtained from the Freescale Literature Distribution center at http://www.freescale.com.

7.4 Unsupported e300c4 Core Features

7.4.1 Instructions

Two Power Architecture instructions are not supported by the module. These two instructions are eciwx and ecowx. The execution of both instructions generates a $\overline{\text{TEA}}$ signal on the CSB. This causes a machine check exception or a checkstop.

7.4.2 CSB Parity

Enabling of the address or data parity error check by setting the HID0[EBA, EBD] bits generates a machine check exception or a checkstop depending on the HID0[EMCP] bit.

Chapter 8
CSB Arbiter and Bus Monitor

8.1 Introduction

This chapter describes the Coherent Systems Bus (CSB) arbiter in the device. In addition, it describes the configuration, control, and status registers of the arbiter.

The CSB arbiter is responsible for providing coherent system bus arbitration. It tracks all the address and data tenures, and provides all the arbitration signals to masters and slaves. In addition, it monitors the bus and reports on errors and protocol violations.

8.1.1 Features

The CSB arbiter includes the following features:

- Supports a programmable pipeline depth (from 1 to 4)
- Supports four levels of priority for bus arbitration
- Supports repeat request mode: number of programmable consecutive transactions from the same master (as many as eight transactions)
- Supports data streaming operations
- Supports programmable address bus parking mode: disable, park to last bus owner, park to s/w selected master
- Claims address only, reserved, and illegal transaction types, report on it and can raise maskable interrupt
- Provides timers for address tenure time-out and data tenure time-out detection and can issue maskable interrupt, if any timer expired
- Reports on transfer error and can issue maskable interrupt
- Can issue regular or machine check interrupt for each type of error event (programmable)

8.1.1.1 Coherent System Bus Overview

Coherent system bus is the central bus. Any data transaction from master to slave in the device passes through the coherent system bus. The coherent system bus supports pipelined transactions. It has independent address and data tenures. Pipeline depth determines the number of address tenures that can be started before the first data tenure is finished.

Basic burst size is equal to cache line length, which is 32 bytes. Using repeat request mode enables as many as eight consecutive bursts to be executed by the same master. The maximum number of consecutive

transactions can be limited by programming the Arbiter Configuration Register (See Section 8.2.1.1, “Arbiter Configuration Register (ACR),” for more details).

8.2 Memory Map/Register Definition

Table 8-1 shows the memory map for arbiter’s configuration, control and status registers.

Table 8-1. Arbiter memory map

Offset from ARB_BASE (0xFF40_0D00)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x00	ACR—Arbiter Configuration Register	R/W	0x0UU0_0000	8.2.1.1/8-176
0x04	ATR—Arbiter Timers Register	R/W	0xFFFF_FFFF	8.2.1.2/8-178
0x08	ATER—Arbiter Transfer Error Register	R/W	0x0000_003F	8.2.1.3/8-178
0x0C	AER—Arbiter Event Register	R/W	0x0000_0000	8.2.1.4/8-179
0x10	AIDR—Arbiter Interrupt Definition Register	R/W	0x0000_0000	8.2.1.5/8-180
0x14	AMR—Arbiter Mask Register	R/W	0x0000_0000	8.2.1.6/8-181
0x18	AEATR—Arbiter Event Attributes Register	R	0x0000_0000	8.2.1.7/8-182
0x1C	AEADR—Arbiter Event Address Register	R	0x0000_0000	8.2.1.8/8-184
0x20	AERR—Arbiter Event Response Register	R/W	0x0000_0000	8.2.1.9/8-185
0x24–0xFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

3 In this column, the symbol “U” indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

8.2.1 Register Descriptions

8.2.1.1 Arbiter Configuration Register (ACR)

The Arbiter Configuration Register (ACR) defines the arbiter modes and parked master on the bus. Figure 8-1 shows the fields of ACR.

Address Base + 0x00 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	DTO_DIS	ATO_DIS	CORE DIS	0	0	AACKWS		0	0	PIPE_DEP	
W																
Reset	0	0	0	0	0	0	0	U[1]	0	0	U[1]	U[1]	0	0	0	0

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

Figure 8-1. Arbiter Configuration Register (ACR)

[1] The reset values of COREDIS and AACKWS are determined from the reset configuration word.

Table 8-2. ACR field descriptions

Field	Description
DTO_DIS	Data time out detection disable. 0 Do not stop the DTO counter and do not prevent data time out detection. 1 Stop the DTO counter and prevent data time out detection.
ATO_DIS	Address time out detection disable. 0 Do not stop the ATO counter and do not prevent address time out detection. 1 Stop the ATO counter and prevent address time out detection.
COREDIS	This bit must always be cleared (set to 0).
AACKWS	Reserved. Write should preserve reset value. Address acknowledge wait states. Specifies minimum number of address tenure wait states. This is the minimum delay between assertion of $\overline{TS}$ and assertion of $\overline{AACK}$. 00 $\overline{AACK}$ is asserted minimum 1 cycle after $\overline{TS}$ 01 $\overline{AACK}$ is asserted minimum 2 cycles after $\overline{TS}$ 10 $\overline{AACK}$ is asserted minimum 3 cycles after $\overline{TS}$ 11 $\overline{AACK}$ is asserted minimum 4 cycles after $\overline{TS}$ **Note:** 00 option can be used only if the processor is not operating in 1:1 or 3:2 bus clock ratios
PIPE_DEP	Pipeline depth (number of outstanding transactions). 000 Pipeline depth 1 (1 outstanding transaction) 001 Pipeline depth 2 (2 outstanding transactions) 010 Pipeline depth 3 (3 outstanding transactions) 011 Pipeline depth 4 (4 outstanding transactions) Others Reserved
RPTCNT	Repeat count. Specifies the maximum number of consecutive transactions, that any master can perform, using $\overline{REPEAT}$ request mode. 000 1 consecutive transactions ($\overline{REPEAT}$ request mode disable) 001 2 consecutive transactions 010 3 consecutive transactions 011 4 consecutive transactions 100 5 consecutive transactions 101 6 consecutive transactions 110 7 consecutive transactions 111 8 consecutive transactions **Note:** It is recommended not to program this field for more than 4 consecutive transactions.
WPARK	WOP Parking. Specifies, whether bus is parked to CPU on WOP cycle (cycle after $\overline{ARTRY}$ assertion). 0 Park to CPU 1 Do not park bus to any master at WOP cycle
APARK	Address parking. Specifies arbiter bus parking mode. 00 Park to master. Arbiter parks the address bus to the master, that is selected by numeric value of PARKM field. 01 Park to last owner. Arbiter parks the address bus to last bus owner. 10 Disable. Arbiter does not assert $\overline{BG}$ to any master, if no $\overline{BR}$ is present. 11 Reserved
PARKM	Parking master. 0000 Power Architecture Core 0001 0010 SAP 0011–1111 Reserved

8.2.1.2 Arbiter Timers Register (ATR)

Arbiter timers register (ATR) defines the arbiter address time out (ATO), data time out (DTO) timer's values. Figure 8-2 shows the fields of ATR.

Address: Base + 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DTO															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ATO															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 8-2. Arbiter Timers Register (ATR)

Table 8-3. ATR field descriptions

Field	Description
DTO	Data time out. Specifies the time-out period for the data tenure. The granularity of this field is 128 bus clocks. The maximum value is 8388480 coherent system bus clocks. Data time_out occurs, if the data tenure was not ended before the specified time-out period timer expires between the assertion of $\overline{\text{DBB}}$ until the assertion of last $\overline{\text{TA}}$. When DTO = *n*, the timeout cycle is *n* × 128. 0x0000 Reserved 0x0001 128 clock cycles 0x0002 256 clock cycles 0x0003 384 clock cycles ... ... 0xFFFF 8388480 clock cycles
ATO	Address time out. Specifies the time-out period for the address tenure. The granularity of this field is 128 bus clocks. Maximum value is 8388480 coherent system bus clocks. Address time-out occurs, if the address tenure was not ended before the specified time-out period timer expires between assertion of $\overline{\text{TS}}$ signal until the assertion of $\overline{\text{AACK}}$ signal. When ATO = *n*, the timeout cycle is *n* × 128. 0x0000 Reserved 0x0001 128 clock cycles 0x0002 256 clock cycles 0x0003 384 clock cycles ... ... 0xFFFF 8388480 clock cycles

8.2.1.3 Arbiter Transfer Error Register (ATER)

The Arbiter Transfer Error Register (ATER) allows defining an event as a non-error event. If an event is defined as a non-error event, an occurrence of the event is not reported in either the Arbiter Event register

(AER) or the event attributes and address register. For transfer types that are not defined as error events, the arbiter also does not end address/data tenures. Figure 8-3 shows the fields of ATER.

Address: Base + 0x08 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ETEA	RES	ECW	AO	DTO	ATO
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 8-3. Arbiter Transfer Error Register (ATER)

Table 8-4. ATER field descriptions

Field	Description
ETEA	External $\overline{\text{TEA}}$. Specifies whether the assertion of the $\overline{\text{TEA}}$ signal by one of the slaves is reported in arbiter event registers. 0 Assertion of $\overline{\text{TEA}}$ signal by one of the slaves is not reported in arbiter event registers. 1 Assertion of $\overline{\text{TEA}}$ signal by one of the slaves is reported in arbiter event registers.
RES	Reserved transfer type. Specifies whether a transaction with a reserved transfer type is reported in arbiter event registers. 0 Reserved transaction is not reported in arbiter event registers. 1 Reserved transaction is reported in arbiter event registers.
ECW	External Control Word transfer type. Specifies whether a transaction with an external control word transfer type is reported in arbiter event registers. 0 External control word read/write transaction is not reported in arbiter event registers. 1 External control word read/write transaction is reported in arbiter event registers.
AO	Address Only transfer type. Specifies whether a transaction with address only transfer type is reported in arbiter event registers. 0 Address only transaction is not reported in arbiter event registers. 1 Address only transaction is reported in arbiter event registers.
DTO	Data Time Out. Specifies whether data tenure time out is reported in arbiter event registers. 0 Data time out is not reported in arbiter event registers. 1 Data time out is reported in arbiter event registers.
ATO	Address Time Out. Specifies whether address tenure time out is reported in arbiter event registers. 0 Address time out is not reported in arbiter event registers. 1 Address time out is reported in arbiter event registers.

8.2.1.4 Arbiter Event Register (AER)

The Arbiter Event Register (AER) reports erroneous transactions. Bits in this register are cleared by writing logic 1’s to the bit locations in the register. Figure 8-4 shows the fields of AER.

Address: Base + 0x0C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ETEA	RES	ECW	AO	DTO	ATO
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-4. Arbiter Event Register (AER)

Table 8-5. AER field descriptions

Field	Description
ETEA	Transfer error. External $\overline{\text{TEA}}$. Reports on detection of transfer error assertion of $\overline{\text{TEA}}$ signal by one of the slaves. 0 No transfer error detected $\overline{\text{TEA}}$ signal is not asserted by one of the slaves. 1 Transfer error detected $\overline{\text{TEA}}$ signal is asserted by one of the slaves.
RES	Reserved transfer type. Reports on transaction with reserved transfer type. See Section 8.3.2.5, “Reserved Transaction Type,” for more information. 0 No transaction with reserved transfer type occurred. 1 Transaction with reserved transfer type occurred.
ECW	External control word transfer type. Reports on transaction with external control word transfer type. See Section 8.3.2.6, “Illegal (ECIWX/ECOWX) Transaction Type,” for more information. 0 No transaction with external control word transfer type occurred. 1 Transaction with external control word transfer type occurred.
AO	Address Only transfer type. Reports on transaction with address only transfer type. See Section 8.3.2.4, “Address-Only Transaction Type,” for more information. 0 No transaction with address only transfer type occurred. 1 Transaction with address only transfer type occurred.
DTO	Data time out. Reports on data tenure time out. 0 Data time out timer is not expired. 1 Data time out timer is expired.
ATO	Address time out. Reports on address tenure time out. 0 Address time out timer is not expired. 1 Address time out timer is expired.

8.2.1.5 Arbiter Interrupt Definition Register (AIDR)

The Arbiter Interrupt Definition Register (AIDR) determines the interrupt that responds to different error conditions. Setting a bit defines the corresponding interrupt as an MCP interrupt ; clearing a bit defines the corresponding interrupt as a regular interrupt. Figure 8-5 shows the fields of AIDR.

Address: Base + 0x10 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ETEA	RES	ECW	AO	DTO	ATO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-5. Arbiter Interrupt Definition Register (AIDR)

Table 8-6. AIDR field descriptions

Field	Description
ETEA	Transfer error. External $\overline{\text{TEA}}$. Detection of transfer error assertion of $\overline{\text{TEA}}$ signal by one of the slaves interrupt definition. 0 Detection of transfer error assertion of $\overline{\text{TEA}}$ signal by one of the slaves causes a regular interrupt. 1 Detection of transfer error assertion of $\overline{\text{TEA}}$ signal by one of the slaves causes an MCP interrupt.
RES	Reserved transfer type. Transaction with reserved transfer type interrupt definition. 0 Transaction with reserved transfer type causes a regular interrupt. 1 Transaction with reserved transfer type causes an MCP interrupt.
ECW	External control word transfer type. Transaction with external control word transfer type interrupt definition. 0 Transaction with external control word transfer type causes a regular interrupt. 1 Transaction with external control word transfer type causes an MCP interrupt.
AO	Address only transfer type. Transaction with address only transfer type interrupt definition. 0 Transaction with address only transfer type causes a regular interrupt. 1 Transaction with address only transfer type causes an MCP interrupt.
DTO	Data time out. Data tenure time out interrupt definition. 0 Data tenure time out causes a regular interrupt. 1 Data tenure time out causes an MCP interrupt.
ATO	Address time out. Address tenure time out interrupt definition. 0 Address tenure time out causes a regular interrupt. 1 Address tenure time out causes an MCP interrupt.

8.2.1.6 Arbiter Mask Register (AMR)

The Arbiter Mask Register (AMR) is used to mask interrupts or reset requests. Setting a mask bit enables the corresponding interrupt or reset request; clearing a bit masks it. Regular interrupts, MCP interrupts, and reset requests can be masked by AMR. Figure 8-6 shows the fields of AMR.

Address: Base + 0x0014 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ETEA	RES	ECW	AO	DTO	ATO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-6. Arbiter Mask Register (AMR)

Table 8-7. AMR field descriptions

Field	Description
ETEA	Transfer error. External $\overline{\text{TEA}}$. Detection of transfer error assertion of $\overline{\text{TEA}}$ signal by one of the slaves interrupt mask bit. 0 Detection of transfer error assertion of $\overline{\text{TEA}}$ signal by one of the slaves interrupt is disabled. 1 Detection of transfer error assertion of $\overline{\text{TEA}}$ signal by one of the slaves interrupt is enabled.
RES	Reserved transfer type.Transaction with reserved transfer type interrupt mask bit. 0 Transaction with reserved transfer type interrupt is disabled. 1 Transaction with reserved transfer type interrupt is enabled.
ECW	External control word transfer type.Transaction with external control word transfer type interrupt mask bit. 0 Transaction with external control word transfer type interrupt is disabled. 1 Transaction with external control word transfer type interrupt is enabled.
AO	Address only transfer type. Transaction with address only transfer type interrupt mask bit. 0 Transaction with address only transfer type interrupt is disabled. 1 Transaction with address only transfer type interrupt is enabled.
DTO	Data time out. Data tenure time out interrupt mask bit. 0 Data tenure time out interrupt is disabled. 1 Data tenure time out interrupt is enabled.
ATO	Address time out. Address tenure time out interrupt mask bit. 0 Address tenure time out interrupt is disabled. 1 Address tenure time out interrupt is enabled.

8.2.1.7 Arbiter Event Attributes Register (AEATR)

The Arbiter Event Attributes Register (AEATR) reports the type of transaction that caused the error that is specified in the Arbiter Event Register (AER). See Section 8.2.1.4, “Arbiter Event Register (AER),” for more information. AEATR is cleared only by power-on reset. The attributes of the first error event are stored. As AEATR is not affected by soft or hard reset, software can read this register and determine the cause of the bus failure, even if the bus is stalled. Figure 8-7 shows the fields of AEATR.

Address: Base + 0x0018 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	EVENT			0	0	0	MSTR_ID				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	TBST	TSIZE			0	0	0	TTYPE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-7. Arbiter Event Attributes Register (AEATR)

Table 8-8. AEATR field descriptions

Field	Description
EVENT	Event type. 000 Address time out. 001 Data time out. 010 Address only transfer type. 011 External control word transfer type. 100 Reserved transfer type. 101 Transfer error External $\overline{\text{TEA}}$. 11x Reserved
MSTR_ID	Master ID. 00000 Power Architecture core data transaction. 00010 Power Architecture core instruction fetch. 01010 JTAG 01011 Reserved Others Reserved **Note:** Master ID reflects the source of the transaction and is used for debug purposes.
TBST	Transfer burst. 0 Transfer size is 8 bytes or smaller. 1 Transfer size is larger than 8 bytes.

Table 8-8. AEATR field descriptions (Continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>TSIZE</td><td>Transfer Size. Transfer size encoding depends on the value of $\overline{\text{TBST}}$.
TBST = 0
<table>
<tr><th>TSIZE</th><th>Transfer size</th><th>TSIZE</th><th>Transfer size</th></tr>
<tr><td>000</td><td>8 bytes</td><td>100</td><td>4 bytes</td></tr>
<tr><td>001</td><td>1 byte</td><td>101</td><td>5 bytes</td></tr>
<tr><td>010</td><td>2 bytes</td><td>110</td><td>6 bytes</td></tr>
<tr><td>011</td><td>3 bytes</td><td>111</td><td>7 bytes</td></tr>
</table>
TBST = 1
<table>
<tr><th>TSIZE</th><th>Transfer size</th><th>TSIZE</th><th>Transfer size</th></tr>
<tr><td>000</td><td>16 bytes</td><td>010</td><td>32 bytes</td></tr>
<tr><td>001</td><td>24 bytes</td><td>all other values</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>TTYPE</td><td>Transfer Type.
<table>
<tr><th>TTYPE</th><th>Transfer type</th><th>TTYPE</th><th>Transfer type</th></tr>
<tr><td>00000</td><td>Address-only</td><td>01101</td><td>Address-only</td></tr>
<tr><td>00001</td><td>Address-only</td><td>01110</td><td>Burst read</td></tr>
<tr><td>00010</td><td>Single beat or burst write</td><td>10000</td><td>Address-only</td></tr>
<tr><td>00100</td><td>Address-only</td><td>10010</td><td>Single beat write</td></tr>
<tr><td>00110</td><td>Burst write</td><td>10100</td><td>ECOWX—Illegal single beat write</td></tr>
<tr><td>01000</td><td>Address-only</td><td>11000</td><td>Address-only</td></tr>
<tr><td>01001</td><td>Address-only</td><td>11010</td><td>Single beat or burst read</td></tr>
<tr><td>01010</td><td>Single beat or burst read</td><td>11100</td><td>ECIWX—Illegal single beat read</td></tr>
<tr><td>01011</td><td>Single beat or burst read</td><td>11110</td><td>Burst read</td></tr>
<tr><td>01100</td><td>Address-only</td><td colspan="2">All other values are reserved.</td></tr>
</table></td></tr>
</table>

8.2.1.8 Arbiter Event Address Register (AEADR)

The Arbiter Event Address Register (AEADR) reports the address of a transaction that caused the error that is specified in the Arbiter Event Register (AER). See Section 8.2.1.4, “Arbiter Event Register (AER),” for more information. AEADR is cleared only by power-on reset. The address of the first error event is stored. As AEADR is not affected by soft or hard reset, software can read this register and determine the cause of the bus failure, even if the bus is stalled.

Figure 8-8 shows the fields of AEADR.

Address: Base + 0x001C Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-8. Arbiter Event Address Register (AEADR)

Table 8-9. AEADR field descriptions

Field	Description
ADDR	Address of the event that was reported in AEATR register. See Section 8.2.1.7, “Arbiter Event Attributes Register (AEATR),” for more information.

8.2.1.9 Arbiter Event Response Register (AERR)

The Arbiter Event Response Register (AERR) determines whether different error conditions cause an interrupt or a reset request. Setting a bit defines the corresponding error condition to cause a reset request; clearing a bit defines the corresponding error condition to cause an interrupt. Figure 8-9 shows the fields of AERR.

Address: Base + 0x0020 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ETEA	RES	ECW	AO	DTO	ATO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-9. Arbiter Event Response Register (AERR)

Table 8-10. AERR field descriptions

Field	Description
ETEA	Transfer error. External $\overline{\text{TEA}}$. Detection of transfer error assertion of $\overline{\text{TEA}}$ signal by one of the slaves event response. 0 Detection of transfer error assertion of $\overline{\text{TEA}}$ signal by one of the slaves causes an interrupt. 1 Detection of transfer error assertion of $\overline{\text{TEA}}$ signal by one of the slaves causes a reset request.

Table 8-10. AERR field descriptions (Continued)

Field	Description
RES	Reserved transfer type. Transaction with reserved transfer type interrupt definition. 0 Transaction with reserved transfer type causes an interrupt. 1 Transaction with reserved transfer type causes a reset request.
ECW	External control word transfer type. Transaction with external control word transfer type interrupt definition. 0 Transaction with external control word transfer type causes an interrupt. 1 Transaction with external control word transfer type causes a reset request.
AO	Address only transfer type. Transaction with address only transfer type interrupt definition. 0 Transaction with address only transfer type causes an interrupt. 1 Transaction with address only transfer type causes a reset request.
DTO	Data time out. Data tenure time out interrupt definition. 0 Data tenure time out causes an interrupt. 1 Data tenure time out causes a reset request.
ATO	Address time out. Address tenure time out interrupt definition. 0 Address tenure time out causes an interrupt. 1 Address tenure time out causes a reset request.

8.3 Functional Description

The following sections describe arbiter functionality: arbitration policy and bus error detection.

8.3.1 Arbitration Policy

The arbitration process involves the masters and the arbiter. The masters arbitrate for the privilege of owning the address tenure. For the data tenure, the CSB arbiter uses the same order of transactions as address tenures. Figure 8-10 shows the interface signals between the CSB arbiter and masters that are involved in the address bus arbitration.

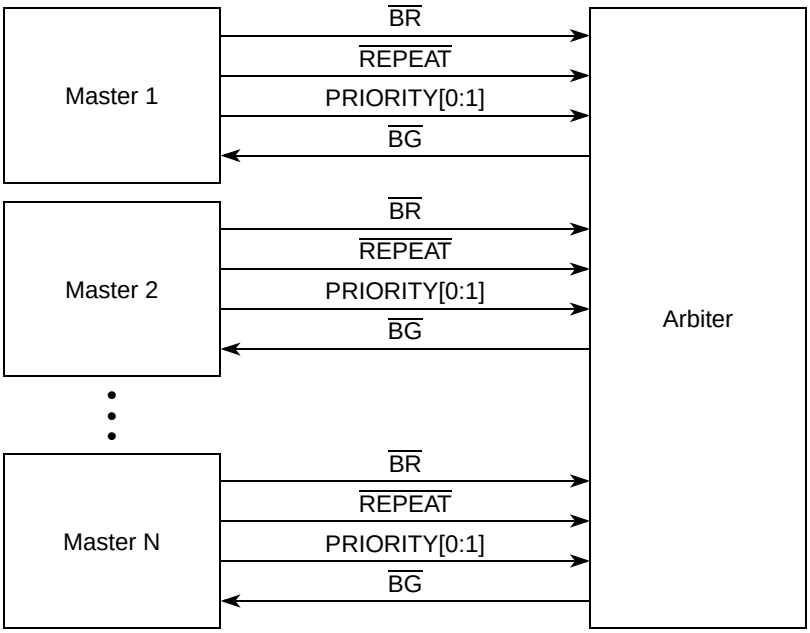


Figure 8-10. Address Bus Arbitration

A master has to acquire the address bus ownership before it starts any transaction. The master asserts its own bus request signal along with the arbitration attribute signals REPEAT and PRIORITY[0:1]. The arbiter later asserts the corresponding address bus grant signal to the requesting master depending on the system states and arbitration scheme. See Section 8.3.1.1, "Address Bus Arbitration With PRIORITY[0:1]," for details information on arbitration scheme. When the address bus grant is received, the master can start the address tenure.

8.3.1.1 Address Bus Arbitration With PRIORITY[0:1]

When a master asserts its bus request to acquire the address bus ownership, it can drive its PRIORITY[0:1] signals to indicate request priority. The master would be served sooner because of its higher priority level. The arbiter takes this extra information into consideration to yield better service for a higher priority request than a lower priority request. Therefore, the arbiter operates according to the following priority based arbitration scheme:

1. For every priority level fair arbitration scheme is used (a simple Round Robin scheme).
2. For every priority level other than 0, one place is reserved as a place holder for lower level arbitration rings.
3. Each master can change its priority level at any time.

Figure 8-11 shows an example of priority based arbitration algorithm with four priority levels. In this example, if all masters request the bus continuously, the following order of bus grants occurs with the specific bandwidth:

- M6 gets 1/2 of the bus bandwidth
- M4 and M5 each get 1/6 of the bus bandwidth
- M0 and M3 each get 1/18 of the bus bandwidth

- M1 and M2 each get 1/36 of the bus bandwidth

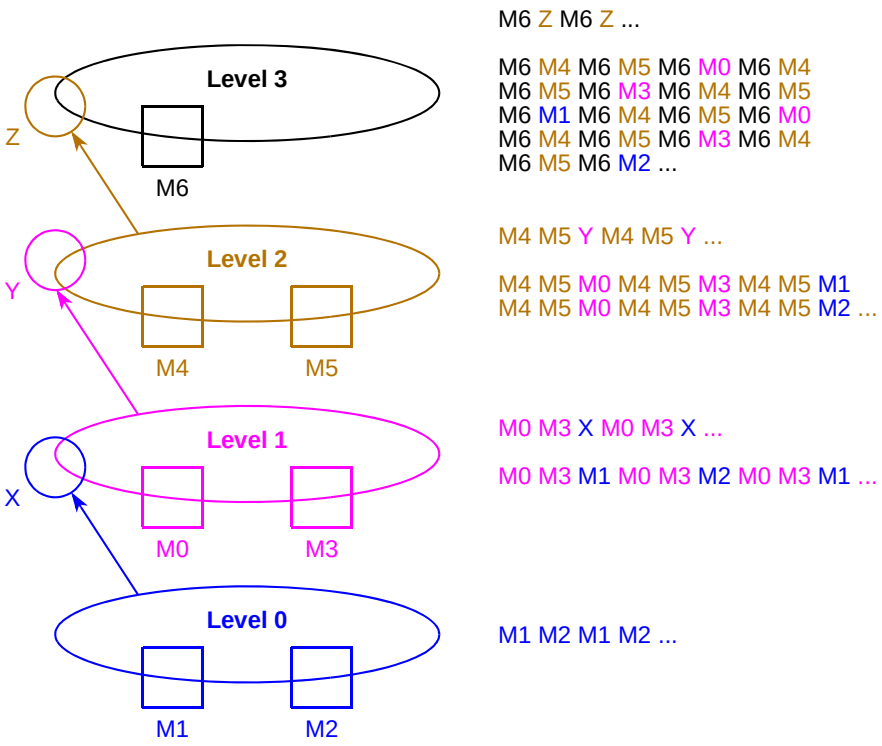


Figure 8-11. Example of Priority Based Arbitration Algorithm

8.3.1.2 Address Bus Arbitration With $\overline{\text{REPEAT}}$

When a master owns the current address bus and wants to perform another transaction, it can assert bus request along with $\overline{\text{REPEAT}}$, to make a repeat request to the arbiter. Consequently, the arbiter asserts bus grant to the same master if the current address tenure is not being $\overline{\text{ARTRY}}$'d. This happens regardless of the priority level of bus request from other masters. In other words, repeat request overrides the priority scheme.

Even though repeat request can improve the page hit ratio and the overall memory bandwidth efficiency, it can increase the worst case latency of individual master. Therefore, the arbiter has a programmable counter to limit the maximum number of consecutive transactions that are performed by masters. When the counter expires, the arbiter ignores the $\overline{\text{REPEAT}}$ signal and falls back to the regular arbitration scheme.

See Section 8.2.1.1, "Arbiter Configuration Register (ACR)," for more details about programming ACR[RPTCNT].

8.3.1.3 Address Bus Arbitration After $\overline{\text{ARTRY}}$

The $\overline{\text{ARTRY}}$ protocol is used primarily by the CPU to interrupt a transaction that hits to a modified line in its D-cache, so that it can maintain data coherency by performing the snoop copyback. In addition, any master and/or slave can $\overline{\text{ARTRY}}$ a transaction for whatever reason.

When an address tenure is $\overline{\text{ARTRY}}$'d, all masters must negate their bus request signals during the cycle after $\overline{\text{ARTRY}}$ except the master that asserted ARTRY signal. The cycle after $\overline{\text{ARTRY}}$ is called WOP (Window Of Opportunity).

During the WOP cycle, the arbiter performs the arbitration the same way as in regular arbitration cycle, except that the parking master policy does not apply. When the CPU asserts $\overline{\text{ARTRY}}$, the bus is immediately granted to the CPU to perform snoop copyback. After the completion of snoop copyback, the arbiter grants the bus back to the master that had its transaction $\overline{\text{ARTRY}}$'d. If $\overline{\text{ARTRY}}$ was asserted by any other master or slave, the arbiter performs the arbitration the same way as it would perform it for a regular arbitration cycle. The master that had its transaction $\overline{\text{ARTRY}}$ed by any master or slave except CPU is put at the end of priority list. It can be programmed to park the bus to CPU on WOP cycle or not to park it to any master. Refer to Section 8.2.1.1, "Arbiter Configuration Register (ACR)," for more detail about ACR[WPARK].

8.3.1.4 Address Bus Parking

The arbiter supports address bus parking. When no master is requesting the bus, the arbiter can grant mastership of the bus to a specified master. This is referred to as bus parking. The parked master can skip the bus request and assume the bus mastership directly. This reduces the access latency for parked master.

See Section 8.2.1.1, "Arbiter Configuration Register (ACR)," for more details about ACR[APARK] and ACR[PARKM].

8.3.1.5 Data Bus Arbitration

For every committed address tenure except address-only or reserved-type transactions, a data tenure is required to complete the transaction.

In the system, the arbiter controls the issuing of data bus grants to a master and a slave, which are involved in a data tenure of a previously performed address tenure.

- In data tenure without data streaming (master or slave or both do not support data streaming), the arbiter guarantees that the data bus grant is asserted only when the data bus is idle.
- In data tenure with data streaming (both master and slave support data streaming), the arbiter guarantees that the data bus grant is asserted when the data bus is either idle or waiting for the last transfer acknowledge.

8.3.2 Bus Error Detection

The arbiter is responsible for tracking the following cases on the bus:

- Address time out
- Data time out
- Transfer error External $\overline{\text{TEA}}$
- Address only transaction type
- Reserved transaction type
- Illegal (ECWIX/ECWOX) transaction type

8.3.2.1 Address Time Out

An address time out occurs if the address tenure was not ended before the specified time-out period (programmed by ATR[ATO]) expires between the assertion of the $\overline{TS}$ signal until the assertion of the $\overline{AACK}$ signal. In this case, the arbiter performs as follows:

1. Ends the address tenure by asserting $\overline{AACK}$.
2. Starts data tenure and ends it by asserting transfer error $\overline{TEA}$.
3. Reports on the event to AER[ATO] if reporting is enabled by ATER[ATO].
4. Issues a reset request, an MCP, or a regular interrupt according to AERR[ATO] and AIDR[ATO] if enabled by AMR[ATO] and if reporting is enabled by ATER[ATO].
5. Updates the transaction attributes and address of AEATR and AEADR for the first error event.

8.3.2.2 Data Time Out

A data time out occurs if the data tenure was not ended before the specified time-out period (programmed by ATR[DTO]) expires between the assertion of $\overline{DBB}$ until the assertion of last $\overline{TA}$. In this case, the arbiter performs as follows:

1. Ends the data tenure by asserting transfer error $\overline{TEA}$.
2. Reports on this event in AER[DTO] if reporting is enabled by ATER[DTO].
3. Issues a reset request, an MCP, or a regular interrupt according to AERR[DTO] and AIDR[DTO], if enabled by AMR[DTO] and if reporting is enabled by ATER[DTO].
4. Updates transaction attributes and address of AEATR and AEADR for the first error event.

8.3.2.3 Transfer error External $\overline{TEA}$

The arbiter tracks the transfer error $\overline{TEA}$ signal that is asserted by one of the slaves. In this case, the arbiter performs as follows:

1. Reports on the event to AER[ETEA] if reporting is enabled by ATER[ETEA].
2. Issues a reset request, an MCP, or a regular interrupt according to AERR[ETEA] and AIDR[ETEA] if enabled by AMR[ETEA] and if reporting is enabled by ATER[ETEA].
3. Updates the transaction attributes and address of AEATR and AEADR for the first error event.

8.3.2.4 Address-Only Transaction Type

Table 8-11 lists the transaction types that are defined as address-only:

Table 8-11. Address-Only Transaction Type Encoding

TTYPE[0:4]	Bus Command
00000	Clean block
00100	Flush block
01000	Sync
01100	Kill block

Table 8-11. Address-Only Transaction Type Encoding

TTYPE[0:4]	Bus Command
10000	eieio
11000	TLB Invalidate
00001	lwarx reservation set
01001	tlbsync
01101	icbi

The arbiter allows address-only (AO) transactions on the bus and the e300 core can issue address-only (AO) transactions (see HID0 [ABE] in the *Power Architecture Core Family Reference Manual*). Because there is no advantage in using AO transactions in this system, the bus monitor allows the detection of AO transactions and treats them as an error.

For transactions with an address-only transfer type, the arbiter performs as follows:

1. Ends the address tenure by asserting $\overline{\text{AACK}}$.
2. Reports on the event to AER[AO] if reporting is enabled by ATER[AO].
3. Issues a reset request, an MCP, or a regular interrupt according to AERR[AO] and AIDR[AO] if enabled by AMR[AO] and if reporting is enabled by ATER[AO].
4. Updates the transaction attributes and address of AEATR and AEADR for the first error event.

8.3.2.5 Reserved Transaction Type

Table 8-12 shows transaction types that are defined as reserved:

Table 8-12. Reserved Transaction Type Encoding

TTYPE[0:4]	Bus Command
00101	Reserved
1XX01	Reserved for customer
10110	Reserved
00011	Reserved
00111	Reserved
01111	Reserved
1XX11	Reserved for customer

For transactions with a reserved transfer type, the arbiter performs as follows:

1. Ends the address tenure by asserting $\overline{\text{AACK}}$.
2. Reports on the event to AER[RES] if reporting is enabled by ATER[RES].
3. Issues a reset request, an MCP, or a regular interrupt according to AERR[RES] and AIDR[RES], if enabled by AMR[RES] and if reporting is enabled by ATER[RES].
4. Updates the transaction attributes and address of AEATR and AEADR for the first error event.

8.3.2.6 Illegal (ECIWX/ECOWX) Transaction Type

Table 8-13 shows transaction types that are defined as illegal.

Table 8-13. Illegal Transaction Type Encoding

TTYPE[0:4]	Bus Command
10100	External control word write (ecowx)
11100	External control word read (eciwx)

For transactions with an illegal (ECIWX, ECOWX) transfer type, the arbiter performs as follows:

1. Ends the address tenure by asserting $\overline{\text{AACK}}$.
2. Starts data tenure and ends data tenure by asserting $\overline{\text{TEA}}$.
3. Reports on the event in AER[ECW] if reporting is enabled by ATER[ECW].
4. Issues a reset request, an MCP, or a regular interrupt according to AERR[ECW] and AIDR[ECW], if enabled by AMR[ECW] and if reporting is enabled by ATER[ECW].
5. Updates transaction attributes and address of AEATR and AEADR for the first error event.

See the following for more information:

- Section 8.2.1.3, “Arbiter Transfer Error Register (ATER)”
- Section 8.2.1.4, “Arbiter Event Register (AER)”
- Section 8.2.1.5, “Arbiter Interrupt Definition Register (AIDR)”
- Section 8.2.1.6, “Arbiter Mask Register (AMR)”
- Section 8.2.1.7, “Arbiter Event Attributes Register (AEATR)”
- Section 8.2.1.8, “Arbiter Event Address Register (AEADR)”
- Section 8.2.1.9, “Arbiter Event Response Register (AERR)”

8.4 Initialization/Applications Information

The following sections describe the initialization and error handling sequences for the arbiter:

8.4.1 Initialization Sequence

The following initialization sequence is recommended:

1. Write to ACR to configure pipeline depth, address bus parking mode, global maximum repeat count, and address acknowledge wait states.
2. Write to ATER to define which events are considered error events and which are not.
3. Write to AERR to define whether different error events cause a reset request or an interrupt.
4. Write to AIDR to define the kind of interrupt (regular or MCP) that is caused by every error event. This is necessary only if interrupts are enabled and AERR defines error events to generate interrupts.
5. Write to AMR to enable interrupts.

6. Write to ATR to set the ATO and DTO timers. This is necessary only if the required timers are set to less than their maximum value (the default).

8.4.2 Error Handling Sequence

The following error handling sequence is recommended:

1. Read AER to find out about the error that occurred in the system. Also, read the values of AEATR and AEADR to check on the first error event in the system.
2. If those registers are not accessible because of a stalled bus, reset the chip and read the values of the AEATR and AEADR registers to check on the event that caused this problem to the system. Use $\overline{\text{HRESET}}$ to reset the chip to guarantee that the information stored in AEATR and AEADR is not lost.
3. Clear all the previous events by writing 1s to AER. This register is also cleared after reset.

This page is intentionally left blank.

Chapter 9
Direct Memory Access (DMA)

9.1 Introduction

The direct memory access (DMA) module provides a flexible and efficient way to move blocks of data within the system. The DMA controller reduces the workload on the microprocessor, allowing it to continue execution of system software.

The DMA module includes a DMA engine, interfaces to peripheral buses and a dynamic clock gating controller. The DMA engine performs source and destination address calculations, actual data movement operations, and has a local memory containing the transfer control descriptors (TCD) for the channels. The interfaces to peripheral buses grab data from source peripherals or pass data to destination peripherals. The dynamic clock gating controller monitors the DMA engine and interface activities, turns off clocks to the DMA engine, and DMA interfaces after detecting that the bus is idle.

9.1.1 Features

The DMA module has the following features:

- Programmable multi-channel interface to peripherals
  - Support for 64 channels with 32-bit data path widths
- Unrestricted data movement within physical memory address space
  - All data movement via dual-address transfers: read from source, write to destination
  - The transfer control descriptors (TCD) support these two-deep, nested transfer operations:
    - Inner data transfer loop defined by a minor byte transfer count
    - Outer data transfer loop defined by a major iteration count
- Support for external DMA request over GPIO interface
  - 31 GPIO pins can be used as DMA requestors
- Flexible protocol programmability
  - Programmable source, destination addresses, and transfer size
  - Transfer control descriptors (TCD) organized to support two-deep, nested transfer operations
  - Support for fixed-priority and round-robin channel arbitration
  - Channel completion reported via interrupt requests
  - Support for scatter/gather DMA processing

9.2 Memory Map and Register Definition

Table 9-1 shows the memory map of the DMA unit.

Table 9-1. DMA memory map

Offset from DMA_BASE (0xFF41_4000)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x0000	DMACR—DMA Control Register	R/W	0x0000_E400	9.2.1.1/9-207
0x0004	DMAES—DMA Error Status	R	0x0000_0000	9.2.1.2/9-208
0x0008	DMAERQH—DMA Enable Request High (Channels 63–32)	R/W	0x0000_0000	9.2.1.3/9-211
0x000C	DMAERQL—DMA Enable Request Low (Channels 31–00)	R/W	0x0000_0000	9.2.1.3/9-211
0x0010	DMAEEIH—DMA Enable Error Interrupt High (Channels 63–32)	R/W	0x0000_0000	9.2.1.4/9-212
0x0014	DMAEEIL—DMA Enable Error Interrupt Low (Channels 31–00)	R/W	0x0000_0000	9.2.1.4/9-212
0x0018	DMASERQ—DMA Set Enable Request	R/W	0x00	9.2.1.5/9-213
0x0019	DMACERQ—DMA Clear Enable Request	R/W	0x00	9.2.1.6/9-213
0x001A	DMASEEI—DMA Set Enable Error Interrupt	R/W	0x00	9.2.1.7/9-214
0x001B	DMACEEI—DMA Clear Enable Error Interrupt	R/W	0x00	9.2.1.8/9-214
0x001C	DMACINT—DMA Clear Interrupt Request	R/W	0x00	9.2.1.9/9-215
0x001D	DMACERR—DMA Clear Error	R/W	0x00	9.2.1.10/9-216
0x001E	DMASSRT—DMA Set Start Bit	R/W	0x00	9.2.1.11/9-216
0x001F	DMACDNE—DMA Clear Done Status Bit	R/W	0x00	9.2.1.12/9-217
0x0020	DMAINTH—DMA Interrupt Request High (Channels 63–32)	R/W	0x0000_0000	9.2.1.13/9-217
0x0024	DMAINTL—DMA Interrupt Request Low (Channels 31–00)	R/W	0x0000_0000	9.2.1.13/9-217
0x0028	DMAERRH—DMA Error High (Channels 63–32)	R/W	0x0000_0000	9.2.1.14/9-218
0x002C	DMAERRL—DMA Error Low (Channels 31–00)	R/W	0x0000_0000	9.2.1.14/9-218
0x0030	DMAHRSH—DMA Hardware Request Status High (Channels 63–32)	R	0x0000_0000	9.2.1.15/9-220
0x0034	DMAHRSL—DMA Hardware Request Status Low (Channels 31–00)	R	0x0000_0000	9.2.1.15/9-220
0x0100	DCHPRI0—DMA Channel 0 Priority	R/W	—[4]	9.2.1.16/9-221
0x0101	DCHPRI1—DMA Channel 1 Priority	R/W	—[4]	9.2.1.16/9-221
0x0102	DCHPRI2—DMA Channel 2 Priority	R/W	—[4]	9.2.1.16/9-221

Table 9-1. DMA memory map (Continued)

Offset from DMA_BASE (0xFF41_4000)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x0103	DCHPRI3—DMA Channel 3 Priority	R/W	—[4]	9.2.1.16/9-221
0x0104	DCHPRI4—DMA Channel 4 Priority	R/W	—[4]	9.2.1.16/9-221
0x0105	DCHPRI5—DMA Channel 5 Priority	R/W	—[4]	9.2.1.16/9-221
0x0106	DCHPRI6—DMA Channel 6 Priority	R/W	—[4]	9.2.1.16/9-221
0x0107	DCHPRI7—DMA Channel 7 Priority	R/W	—[4]	9.2.1.16/9-221
0x0108	DCHPRI8—DMA Channel 8 Priority	R/W	—[4]	9.2.1.16/9-221
0x0109	DCHPRI9—DMA Channel 9 Priority	R/W	—[4]	9.2.1.16/9-221
0x010A	DCHPRI10—DMA Channel 10 Priority	R/W	—[4]	9.2.1.16/9-221
0x010B	DCHPRI11—DMA Channel 11 Priority	R/W	—[4]	9.2.1.16/9-221
0x010C	DCHPRI12—DMA Channel 12 Priority	R/W	—[4]	9.2.1.16/9-221
0x010D	DCHPRI13—DMA Channel 13 Priority	R/W	—[4]	9.2.1.16/9-221
0x010E	DCHPRI14—DMA Channel 14 Priority	R/W	—[4]	9.2.1.16/9-221
0x010F	DCHPRI15—DMA Channel 15 Priority	R/W	—[4]	9.2.1.16/9-221
0x0110	DCHPRI16—DMA Channel 16 Priority	R/W	—[4]	9.2.1.16/9-221
0x0111	DCHPRI17—DMA Channel 17 Priority	R/W	—[4]	9.2.1.16/9-221
0x0112	DCHPRI18—DMA Channel 18 Priority	R/W	—[4]	9.2.1.16/9-221
0x0113	DCHPRI19—DMA Channel 19 Priority	R/W	0x00	9.2.1.16/9-221
0x0114	DCHPRI20—DMA Channel 20 Priority	R/W	—[4]	9.2.1.16/9-221
0x0115	DCHPRI21—DMA Channel 21 Priority	R/W	—[4]	9.2.1.16/9-221

Table 9-1. DMA memory map (Continued)

Offset from DMA_BASE (0xFF41_4000)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x0116	DCHPRI22—DMA Channel 22 Priority	R/W	—[4]	9.2.1.16/9-221
0x0117	DCHPRI23—DMA Channel 23 Priority	R/W	—[4]	9.2.1.16/9-221
0x0118	DCHPRI24—DMA Channel 24 Priority	R/W	—[4]	9.2.1.16/9-221
0x0119	DCHPRI25—DMA Channel 25 Priority	R/W	—[4]	9.2.1.16/9-221
0x011A	DCHPRI26—DMA Channel 26 Priority	R/W	—[4]	9.2.1.16/9-221
0x011B	DCHPRI27—DMA Channel 27 Priority	R/W	—[4]	9.2.1.16/9-221
0x011C	DCHPRI28—DMA Channel 28 Priority	R/W	—[4]	9.2.1.16/9-221
0x011D	DCHPRI29—DMA Channel 29 Priority	R/W	—[4]	9.2.1.16/9-221
0x011E	DCHPRI30—DMA Channel 30 Priority	R/W	—[4]	9.2.1.16/9-221
0x011F	DCHPRI31—DMA Channel 31 Priority	R/W	—[4]	9.2.1.16/9-221
0x0120	DCHPRI32—DMA Channel 32 Priority	R/W	—[4]	9.2.1.16/9-221
0x0121	DCHPRI33—DMA Channel 33 Priority	R/W	—[4]	9.2.1.16/9-221
0x0122	DCHPRI34—DMA Channel 34 Priority	R/W	—[4]	9.2.1.16/9-221
0x0123	DCHPRI35—DMA Channel 35 Priority	R/W	—[4]	9.2.1.16/9-221
0x0124	DCHPRI36—DMA Channel 36 Priority	R/W	—[4]	9.2.1.16/9-221
0x0125	DCHPRI37—DMA Channel 37 Priority	R/W	—[4]	9.2.1.16/9-221
0x0126	DCHPRI38—DMA Channel 38 Priority	R/W	—[4]	9.2.1.16/9-221
0x0127	DCHPRI39—DMA Channel 39 Priority	R/W	—[4]	9.2.1.16/9-221
0x0128	DCHPRI40—DMA Channel 40 Priority	R/W	—[4]	9.2.1.16/9-221

Table 9-1. DMA memory map (Continued)

Offset from DMA_BASE (0xFF41_4000)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x0129	DCHPRI41—DMA Channel 41 Priority	R/W	—[4]	9.2.1.16/9-221
0x012A	DCHPRI42—DMA Channel 42 Priority	R/W	—[4]	9.2.1.16/9-221
0x012B	DCHPRI43—DMA Channel 43 Priority	R/W	—[4]	9.2.1.16/9-221
0x012C	DCHPRI44—DMA Channel 44 Priority	R/W	—[4]	9.2.1.16/9-221
0x012D	DCHPRI45—DMA Channel 45 Priority	R/W	—[4]	9.2.1.16/9-221
0x012E	DCHPRI46—DMA Channel 46 Priority	R/W	—[4]	9.2.1.16/9-221
0x012F	DCHPRI47—DMA Channel 47 Priority	R/W	—[4]	9.2.1.16/9-221
0x0130	DCHPRI48—DMA Channel 48 Priority	R/W	—[4]	9.2.1.16/9-221
0x0131	DCHPRI49—DMA Channel 49 Priority	R/W	—[4]	9.2.1.16/9-221
0x0132	DCHPRI50—DMA Channel 50 Priority	R/W	—[4]	9.2.1.16/9-221
0x0133	DCHPRI51—DMA Channel 51 Priority	R/W	0x00	9.2.1.16/9-221
0x0134	DCHPRI52—DMA Channel 52 Priority	R/W	—[4]	9.2.1.16/9-221
0x0135	DCHPRI53—DMA Channel 53 Priority	R/W	—[4]	9.2.1.16/9-221
0x0136	DCHPRI54—DMA Channel 54 Priority	R/W	—[4]	9.2.1.16/9-221
0x0137	DCHPRI55—DMA Channel 55 Priority	R/W	—[4]	9.2.1.16/9-221
0x0138	CHPRI56D—DMA Channel 56 Priority)	R/W	—[4]	9.2.1.16/9-221
0x0139	DCHPRI57—DMA Channel 57 Priority	R/W	—[4]	9.2.1.16/9-221
0x013A	DCHPRI58—DMA Channel 58 Priority	R/W	—[4]	9.2.1.16/9-221
0x013B	DCHPRI59—DMA Channel 59 Priority	R/W	—[4]	9.2.1.16/9-221

Table 9-1. DMA memory map (Continued)

Offset from DMA_BASE (0xFF41_4000)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x013C	DCHPRI60—DMA Channel 60 Priority	R/W	—[4]	9.2.1.16/9-221
0x013D	DCHPRI61—DMA Channel 61 Priority	R/W	—[4]	9.2.1.16/9-221
0x013E	DCHPRI62—DMA Channel 62 Priority	R/W	—[4]	9.2.1.16/9-221
0x013F	DCHPRI63—DMA Channel 63 Priority	R/W	—[4]	9.2.1.16/9-221
0x0140–0x0FFC	Reserved			
0x1000–0x101C	TCD00—Transfer Control Descriptors Channel 00	R/W	—[4]	9.2.1.17/9-221
0x1020–0x103C	TCD01—Transfer Control Descriptors Channel 01	R/W	—[4]	9.2.1.17/9-221
0x1040–0x105C	TCD02—Transfer Control Descriptors Channel 02	R/W	—[4]	9.2.1.17/9-221
0x1060–0x107C	TCD03—Transfer Control Descriptors Channel 03	R/W	—[4]	9.2.1.17/9-221
0x1080–0x109C	TCD04—Transfer Control Descriptors Channel 04	R/W	—[4]	9.2.1.17/9-221
0x10A0–0x10BC	TCD05—Transfer Control Descriptors Channel 05	R/W	—[4]	9.2.1.17/9-221
0x10C0–0x10DC	TCD06—Transfer Control Descriptors Channel 06	R/W	—[4]	9.2.1.17/9-221
0x10E0–0x10FC	TCD07—Transfer Control Descriptors Channel 07	R/W	—[4]	9.2.1.17/9-221
0x1100–0x113C	TCD08—Transfer Control Descriptors Channel 08	R/W	—[4]	9.2.1.17/9-221
0x1120–0x113C	TCD09—Transfer Control Descriptors Channel 09	R/W	—[4]	9.2.1.17/9-221
0x1140–0x115C	TCD10—Transfer Control Descriptors Channel 10	R/W	—[4]	9.2.1.17/9-221
0x1160–0x117C	TCD11—Transfer Control Descriptors Channel 11	R/W	—[4]	9.2.1.17/9-221
0x1180–0x119C	TCD12—Transfer Control Descriptors Channel 12	R/W	—[4]	9.2.1.17/9-221
0x11A0–0x11BC	TCD13—Transfer Control Descriptors Channel 13	R/W	—[4]	9.2.1.17/9-221
0x11C0–0x11DC	TCD14—Transfer Control Descriptors Channel 14	R/W	—[4]	9.2.1.17/9-221

Table 9-1. DMA memory map (Continued)

Offset from DMA_BASE (0xFF41_4000)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x11E0–0x11FC	TCD15—Transfer Control Descriptors Channel 15	R/W	—[4]	9.2.1.17/9-221
0x1200–0x121C	TCD16—Transfer Control Descriptors Channel 16	R/W	—[4]	9.2.1.17/9-221
0x1220–0x123C	TCD17—Transfer Control Descriptors Channel 17	R/W	—[4]	9.2.1.17/9-221
0x1240–0x125C	TCD18—Transfer Control Descriptors Channel 18	R/W	—[4]	9.2.1.17/9-221
0x1260–0x127C	TCD19—Transfer Control Descriptors Channel 19	R/W	—[4]	9.2.1.17/9-221
0x1280–0x129C	TCD20—Transfer Control Descriptors Channel 20	R/W	—[4]	9.2.1.17/9-221
0x12A0–0x12BC	TCD21—Transfer Control Descriptors Channel 21	R/W	—[4]	9.2.1.17/9-221
0x12C0–0x12DC	TCD22—Transfer Control Descriptors Channel 22	R/W	—[4]	9.2.1.17/9-221
0x12E0–0x12FC	TCD23—Transfer Control Descriptors Channel 23	R/W	—[4]	9.2.1.17/9-221
0x1300–0x133C	TCD24—Transfer Control Descriptors Channel 24	R/W	—[4]	9.2.1.17/9-221
0x1320–0x133C	TCD25—Transfer Control Descriptors Channel 25	R/W	—[4]	9.2.1.17/9-221
0x1340–0x135C	TCD26—Transfer Control Descriptors Channel 26	R/W	—[4]	9.2.1.17/9-221
0x1360–0x137C	TCD27—Transfer Control Descriptors Channel 27	R/W	—[4]	9.2.1.17/9-221
0x1380–0x139C	TCD28—Transfer Control Descriptors Channel 28	R/W	—[4]	9.2.1.17/9-221
0x13A0–0x13BC	TCD29—Transfer Control Descriptors Channel 29	R/W	—[4]	9.2.1.17/9-221
0x13C0–0x13DC	TCD30—Transfer Control Descriptors Channel 30	R/W	—[4]	9.2.1.17/9-221
0x13E0–0x13FC	TCD31—Transfer Control Descriptors Channel 31	R/W	—[4]	9.2.1.17/9-221
0x1400–0x141C	TCD32—Transfer Control Descriptors Channel 32	R/W	—[4]	9.2.1.17/9-221
0x1420–0x143C	TCD33—Transfer Control Descriptors Channel 33	R/W	—[4]	9.2.1.17/9-221

Table 9-1. DMA memory map (Continued)

Offset from DMA_BASE (0xFF41_4000)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x1440–0x145C	TCD34—Transfer Control Descriptors Channel 34	R/W	—[4]	9.2.1.17/9-221
0x1460–0x147C	TCD35—Transfer Control Descriptors Channel 35	R/W	—[4]	9.2.1.17/9-221
0x1480–0x149C	TCD36—Transfer Control Descriptors Channel 36	R/W	—[4]	9.2.1.17/9-221
0x14A0–0x14BC	TCD37—Transfer Control Descriptors Channel 37	R/W	—[4]	9.2.1.17/9-221
0x14C0–0x14DC	TCD38—Transfer Control Descriptors Channel 38	R/W	—[4]	9.2.1.17/9-221
0x15E0–0x14FC	TCD39—Transfer Control Descriptors Channel 39	R/W	—[4]	9.2.1.17/9-221
0x1500–0x153C	TCD40—Transfer Control Descriptors Channel 40	R/W	—[4]	9.2.1.17/9-221
0x1520–0x153C	TCD41—Transfer Control Descriptors Channel 41	R/W	—[4]	9.2.1.17/9-221
0x1540–0x155C	TCD42—Transfer Control Descriptors Channel 42	R/W	—[4]	9.2.1.17/9-221
0x1560–0x157C	TCD43—Transfer Control Descriptors Channel 43	R/W	—[4]	9.2.1.17/9-221
0x1580–0x159C	TCD44—Transfer Control Descriptors Channel 44	R/W	—[4]	9.2.1.17/9-221
0x15A0–0x115BC	TCD45—Transfer Control Descriptors Channel 45	R/W	—[4]	9.2.1.17/9-221
0x15C0–0x15DC	TCD46—Transfer Control Descriptors Channel 46	R/W	—[4]	9.2.1.17/9-221
0x15E0–0x15FC	TCD47—Transfer Control Descriptors Channel 47	R/W	—[4]	9.2.1.17/9-221
0x1600–0x161C	TCD48—Transfer Control Descriptors Channel 48	R/W	—[4]	9.2.1.17/9-221
0x1620–0x163C	TCD49—Transfer Control Descriptors Channel 49	R/W	—[4]	9.2.1.17/9-221
0x1640–0x165C	TCD50—Transfer Control Descriptors Channel 50	R/W	—[4]	9.2.1.17/9-221
0x1660–0x167C	TCD51—Transfer Control Descriptors Channel 51	R/W	—[4]	9.2.1.17/9-221
0x1680–0x169C	TCD52—Transfer Control Descriptors Channel 52	R/W	—[4]	9.2.1.17/9-221

Table 9-1. DMA memory map (Continued)

Offset from DMA_BASE (0xFF41_4000)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x16A0–0x16BC	TCD53—Transfer Control Descriptors Channel 53	R/W	—[4]	9.2.1.17/9-221
0x16C0–0x16DC	TCD54—Transfer Control Descriptors Channel 54	R/W	—[4]	9.2.1.17/9-221
0x16E0–0x16FC	TCD55—Transfer Control Descriptors Channel 55	R/W	—[4]	9.2.1.17/9-221
0x1700–0x173C	TCD56—Transfer Control Descriptors Channel 56	R/W	—[4]	9.2.1.17/9-221
0x1720–0x1713C	TCD57—Transfer Control Descriptors Channel 57	R/W	—[4]	9.2.1.17/9-221
0x1740–0x175C	TCD58—Transfer Control Descriptors Channel 58	R/W	—[4]	9.2.1.17/9-221
0x1760–0x177C	TCD59—Transfer Control Descriptors Channel 59	R/W	—[4]	9.2.1.17/9-221
0x1780–0x179C	TCD60—Transfer Control Descriptors Channel 60	R/W	—[4]	9.2.1.17/9-221
0x17A0–0x17BC	TCD61—Transfer Control Descriptors Channel 61	R/W	—[4]	9.2.1.17/9-221
0x117C0–0x171DC	TCD62—Transfer Control Descriptors Channel 62	R/W	—[4]	9.2.1.17/9-221
0x17E0–0x17FC	TCD63—Transfer Control Descriptors Channel 63	R/W	—[4]	9.2.1.17/9-221
0x1800–0x3FFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

3 In this column, the symbol “U” indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

4 Reset value is indeterminate. See register description for more information.

Table 9-2. DMA Block Memory Map

Offset	Register	Access	Section/Page
0x0000	DMACR—DMA Control Register	R/W	9.2.1.1/9-207
0x0004	DMAES—DMA Error Status	R	9.2.1.2/9-208
0x0008	DMAERQH—DMA Enable Request High (Channels 63–32)	R/W	9.2.1.3/9-211
0x000C	DMAERQL—DMA Enable Request Low (Channels 31–00)	R/W	9.2.1.3/9-211
0x0010	DMAEEIH—DMA Enable Error Interrupt High (Channels 63–32)	R/W	9.2.1.4/9-212
0x0014	DMAEEIL—DMA Enable Error Interrupt Low (Channels 31–00)	R/W	9.2.1.4/9-212

Table 9-2. DMA Block Memory Map (Continued)

Offset	Register				Access	Section/Page
0x0018	DMASERQ—DMA Set Enable Request	DMACERQ—DMA Clear Enable Request	DMASEEI—DMA Set Enable Error Interrupt	DMACEEI—DMA Clear Enable Error Interrupt	R/W	9.2.1.5/9-213, 9.2.1.6/9-213, 9.2.1.7/9-214, 9.2.1.8/9-214
0x001C	DMACINT—DMA Clear Interrupt Request	DMACERR—DMA Clear Error	DMASSRT—DMA Set Start Bit	DMACDNE—DMA Clear Done Status Bit	R/W	9.2.1.9/9-215, 9.2.1.10/9-216, 9.2.1.11/9-216, 9.2.1.12/9-217
0x0020	DMAINTH—DMA Interrupt Request High (Channels 63–32)				R/W	9.2.1.13/9-217
0x0024	DMAINTL—DMA Interrupt Request Low (Channels 31–00)				R/W	9.2.1.13/9-217
0x0028	DMAERRH—DMA Error High (Channels 63–32)				R/W	9.2.1.14/9-218
0x002C	DMAERRL—DMA Error Low (Channels 31–00)				R/W	9.2.1.14/9-218
0x0030	DMAHRSH—DMA Hardware Request Status High (Channels 63–32)				R	9.2.1.15/9-220
0x0034	DMAHRSL—DMA Hardware Request Status Low (Channels 31–00)				R	9.2.1.15/9-220
0x0040–0x00FC	Reserved					
0x0100	DCHPRI0—DMA Channel 0 Priority	DCHPRI1—DMA Channel 1 Priority	DCHPRI2—DMA Channel 2 Priority	DCHPRI3—DMA Channel 3 Priority	R/W	9.2.1.16/9-221
0x0104	DCHPRI4—DMA Channel 4 Priority	DCHPRI5—DMA Channel 5 Priority	DCHPRI6—DMA Channel 6 Priority	DCHPRI7—DMA Channel 7 Priority	R/W	9.2.1.16/9-221
0x0108	DCHPRI8—DMA Channel 8 Priority	DCHPRI9—DMA Channel 9 Priority	DCHPRI10—DMA Channel 10 Priority	DCHPRI11—DMA Channel 11 Priority	R/W	9.2.1.16/9-221
0x010C	DCHPRI12—DMA Channel 12 Priority	DCHPRI13—DMA Channel 13 Priority	DCHPRI14—DMA Channel 14 Priority	DCHPRI15—DMA Channel 15 Priority	R/W	9.2.1.16/9-221
0x0110	DCHPRI16—DMA Channel 16 Priority	DCHPRI17—DMA Channel 17 Priority	DCHPRI18—DMA Channel 18 Priority	DCHPRI19—DMA Channel 19 Priority	R/W	9.2.1.16/9-221
0x0114	DCHPRI20—DMA Channel 20 Priority	DCHPRI21—DMA Channel 21 Priority	DCHPRI22—DMA Channel 22 Priority	DCHPRI23—DMA Channel 23 Priority	R/W	9.2.1.16/9-221
0x0118	DCHPRI24—DMA Channel 24 Priority	DCHPRI25—DMA Channel 25 Priority	DCHPRI26—DMA Channel 26 Priority	DCHPRI27—DMA Channel 27 Priority	R/W	9.2.1.16/9-221
0x011C	DCHPRI28—DMA Channel 28 Priority	DCHPRI29—DMA Channel 29 Priority	DCHPRI30—DMA Channel 30 Priority	DCHPRI31—DMA Channel 31 Priority	R/W	9.2.1.16/9-221
0x0120	DCHPRI32—DMA Channel 32 Priority	DCHPRI33—DMA Channel 33 Priority	DCHPRI34—DMA Channel 34 Priority	DCHPRI35—DMA Channel 35 Priority	R/W	9.2.1.16/9-221
0x0124	DCHPRI36—DMA Channel 36 Priority	DCHPRI37—DMA Channel 37 Priority	DCHPRI38—DMA Channel 38 Priority	DCHPRI39—DMA Channel 39 Priority	R/W	9.2.1.16/9-221

Table 9-2. DMA Block Memory Map (Continued)

Offset	Register				Access	Section/Page
0x0128	DCHPRI40—DMA Channel 40 Priority	DCHPRI41—DMA Channel 41 Priority	DCHPRI42—DMA Channel 42 Priority	DCHPRI43—DMA Channel 43 Priority	R/W	9.2.1.16/9-221
0x012C	DCHPRI44—DMA Channel 44 Priority	DCHPRI45—DMA Channel 45 Priority	DCHPRI46—DMA Channel 46 Priority	DCHPRI47—DMA Channel 47 Priority	R/W	9.2.1.16/9-221
0x0130	DCHPRI48—DMA Channel 48 Priority	DCHPRI49—DMA Channel 49 Priority	DCHPRI50—DMA Channel 50 Priority	DCHPRI51—DMA Channel 51 Priority	R/W	9.2.1.16/9-221
0x0134	DCHPRI52—DMA Channel 52 Priority	DCHPRI53—DMA Channel 53 Priority	DCHPRI54—DMA Channel 54 Priority	DCHPRI55—DMA Channel 55 Priority	R/W	9.2.1.16/9-221
0x0138	CHPRI56D—DMA Channel 56 Priority)	DCHPRI57—DMA Channel 57 Priority	DCHPRI58—DMA Channel 58 Priority	DCHPRI59—DMA Channel 59 Priority	R/W	9.2.1.16/9-221
0x013C	DCHPRI60—DMA Channel 60 Priority	DCHPRI61—DMA Channel 61 Priority	DCHPRI62—DMA Channel 62 Priority	DCHPRI63—DMA Channel 63 Priority	R/W	9.2.1.16/9-221
0x0140–0x0FFC	Reserved					
0x1000–0x11FC	TCD00–TCD15				R/W	9.2.1.17/9-221
0x1200–0x13FC	TCD16–TCD31				R/W	9.2.1.17/9-221
0x1400–0x15FC	TCD32–TCD47				R/W	9.2.1.17/9-221
0x1600–0x17FC	TCD48–TCD63				R/W	9.2.1.17/9-221
0x1800–0x3FFF	Reserved					

Table 9-3. DMA Channel Assignments

Requester	DMA Channel	DMA Request Enable	DMA Interrupt Request	DMA Error
GPIO00	DMA_REQ63	ERQ63	INT63	ERR63
GPIO01	DMA_REQ62	ERQ62	INT62	ERR62
GPIO02	DMA_REQ61	ERQ61	INT61	ERR61
GPIO03	DMA_REQ60	ERQ60	INT60	ERR60
GPIO04	DMA_REQ59	ERQ59	INT59	ERR59
GPIO05	DMA_REQ58	ERQ58	INT58	ERR58
GPIO06	DMA_REQ57	ERQ57	INT57	ERR57
GPIO07	DMA_REQ56	ERQ56	INT56	ERR56
GPIO08	DMA_REQ55	ERQ55	INT55	ERR55

Table 9-3. DMA Channel Assignments (Continued)

Requester	DMA Channel	DMA Request Enable	DMA Interrupt Request	DMA Error
GPIO09	DMA_REQ54	ERQ54	INT54	ERR54
GPIO10	DMA_REQ53	ERQ53	INT53	ERR53
GPIO11	DMA_REQ52	ERQ52	INT52	ERR52
GPIO12	DMA_REQ51	ERQ51	INT51	ERR51
GPIO13	DMA_REQ50	ERQ50	INT50	ERR50
GPIO14	DMA_REQ49	ERQ49	INT49	ERR49
GPIO15	DMA_REQ48	ERQ48	INT48	ERR48
GPIO16	DMA_REQ47	ERQ47	INT47	ERR47
GPIO17	DMA_REQ46	ERQ46	INT46	ERR46
GPIO18	DMA_REQ45	ERQ45	INT45	ERR45
GPIO19	DMA_REQ44	ERQ44	INT44	ERR44
GPIO20	DMA_REQ43	ERQ43	INT43	ERR43
GPIO21	DMA_REQ42	ERQ42	INT42	ERR42
GPIO22	DMA_REQ41	ERQ41	INT41	ERR41
GPIO23	DMA_REQ40	ERQ40	INT40	ERR40
GPIO24	DMA_REQ39	ERQ39	INT39	ERR39
GPIO25	DMA_REQ38	ERQ38	INT38	ERR38
GPIO26	DMA_REQ37	ERQ37	INT37	ERR37
GPIO27	DMA_REQ36	ERQ36	INT36	ERR36
GPIO28	DMA_REQ35	ERQ35	INT35	ERR35
GPIO29	DMA_REQ34	ERQ34	INT34	ERR34
GPIO30	DMA_REQ33	ERQ33	INT33	ERR33
MDDRC	DMA_REQ32	ERQ32	INT32	ERR32
LPC	DMA_REQ26	ERQ26	INT26	ERR26
PSC_FIFO_TX9	DMA_REQ21	ERQ21	INT21	ERR21
PSC_FIFO_TX8	DMA_REQ20	ERQ20	INT20	ERR20
PSC_FIFO_TX7	DMA_REQ19	ERQ19	INT19	ERR19
PSC_FIFO_TX6	DMA_REQ18	ERQ18	INT18	ERR18
PSC_FIFO_TX5	DMA_REQ17	ERQ17	INT17	ERR17
PSC_FIFO_TX4	DMA_REQ16	ERQ16	INT16	ERR16
PSC_FIFO_TX3	DMA_REQ15	ERQ15	INT15	ERR15
PSC_FIFO_TX2	DMA_REQ14	ERQ14	INT14	ERR14
PSC_FIFO_TX1	DMA_REQ13	ERQ13	INT13	ERR13

Table 9-3. DMA Channel Assignments (Continued)

Requester	DMA Channel	DMA Request Enable	DMA Interrupt Request	DMA Error
PSC_FIFO_TX0	DMA_REQ12	ERQ12	INT12	ERR12
PSC_FIFO_RX9	DMA_REQ9	ERQ9	INT9	ERR9
PSC_FIFO_RX8	DMA_REQ8	ERQ8	INT8	ERR8
PSC_FIFO_RX7	DMA_REQ7	ERQ7	INT7	ERR7
PSC_FIFO_RX6	DMA_REQ6	ERQ6	INT6	ERR6
PSC_FIFO_RX5	DMA_REQ5	ERQ5	INT5	ERR5
PSC_FIFO_RX4	DMA_REQ4	ERQ4	INT4	ERR4
PSC_FIFO_RX3	DMA_REQ3	ERQ3	INT3	ERR3
PSC_FIFO_RX2	DMA_REQ2	ERQ2	INT2	ERR2
PSC_FIFO_RX1	DMA_REQ1	ERQ1	INT1	ERR1
PSC_FIFO_RX0	DMA_REQ0	ERQ0	INT0	ERR0

9.2.1 Register Descriptions

9.2.1.1 DMA Control Register (DMACR)

The 32-bit DMA control register (DMACR) defines the basic operating configuration of the DMA.

The DMA arbitrates channel service requests in groups of 16 channels. The 64 channel configurations have four groups (3, 2, 1, and 0). Group 3 contains channels 63–48. Group 2 contains channels 47–32. Group 1 contains channels 31–16. Group 0 contains channels 15–0.

Arbitration within a group can be configured to use a fixed priority or a round robin. In fixed priority arbitration, the highest priority channel requesting service is selected to execute. The priorities are assigned by the channel priority registers. In round robin arbitration mode, the channel priorities are ignored and the channels within each group are cycled through without regard to priority.

The group priorities operate in a similar fashion. In group fixed priority arbitration mode, channel service requests in the highest priority group are executed first where priority level 3 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRPnPri registers. All group priorities must have unique values before any channel service requests occur. Otherwise, a configuration error is reported. Unused group priority registers, per configuration, are unimplemented in the DMACR. In group round robin mode, the group priorities are ignored and the groups are cycled through without regard to priority.

Address: Base + 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EDCG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reset	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 9-1. DMA Control Register (DMACR)

Table 9-4. DMACR field descriptions

Field	Description
EDCG	Enable Clock Dynamic Gating 0 Disable clock dynamic gating. 1 Enable clock dynamic gating.
GRP3PRI	Channel Group 3 Priority. Group 3 priority level when fixed priority group arbitration is enabled.
GRP2PRI	Channel Group 2 Priority. Group 2 priority level when fixed priority group arbitration is enabled.
GRP1PRI	Channel Group 1 Priority. Group 1 priority level when fixed priority group arbitration is enabled.
GRP0PRI	Channel Group 0 Priority. Group 0 priority level when fixed priority group arbitration is enabled.
ERGA	Enable Round Robin Group Arbitration 0 Fixed priority arbitration is used for selection among the groups. 1 Round robin arbitration is used for selection among the groups.
ERCA	Enable Round Robin Channel Arbitration 0 Fixed priority arbitration is used for channel selection within each group. 1 Round robin arbitration is used for channel selection within each group.

9.2.1.2 DMA Error Status (DMAES)

The DMA Error Status (DMAES) register provides information concerning the last recorded channel error. Channel errors can be caused by a configuration error (an illegal setting in the transfer control descriptor or an illegal priority register setting in fixed arbitration mode) or an error termination to a bus master read or write cycle.

A configuration error is caused when the starting source or destination address, source or destination offsets, minor loop byte count and the transfer size represent an inconsistent state. The addresses and offsets must be aligned on transfer_size boundaries, and the minor loop byte count must be a multiple of the source and destination transfer sizes. All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively. In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal within a group, or any group priority levels being equal among the groups. All channel priority levels within a group must be unique and all group priority levels among the groups must be unique when fixed arbitration mode is enabled. If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary. If minor loop channel linking is enabled

upon channel completion, a configuration error is reported when the link is attempted if the TCD.CITER.E_LINK bit does not equal the TCD.BITER.E_LINK bit. All configuration error conditions except scatter/gather and minor loop link error are reported as the channel is activated and asserts an error interrupt request, if enabled. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the DMA_ENGINE with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error occurs, the channel is terminated after the read or write transaction already pipelined after errant access, has completed. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel is terminated due to the destination bus error.

The occurrence of any type of error causes the DMA_ENGINE to immediately stop, and the appropriate channel bit in the DMA error register to be asserted. At the same time, the details of the error condition are loaded into the DMAES register. The major loop complete indicators, setting the transfer control descriptor done flag and the possible assertion of an interrupt request, are not affected when an error is detected.

Address: Base + 0x0004 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	ERRCHN[5:0]						SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-2. DMA Error Status Register (DMAES)

Table 9-5. DMAES field descriptions

Field	Description
VLD	Logical OR of all DMAERRH and DMAERRL status bits. 0 No DMAERR bits are set. 1 At least one DMAERR bit is set indicating a valid error exists that has not been cleared.
GPE	Group Priority Error 0 No group priority error. 1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique.
CPE	Channel Priority Error 0 No channel priority error. 1 The last recorded error was a configuration error in the channel priorities within a group. All channel priorities within a group are not unique.
ERRCHN[5:0]	Error Channel Number The channel number of the last recorded error (excluding GPE and CPE errors).
SAE	Source Address Error 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCD.SADDR field. TCD.SADDR is inconsistent with TCD.SSIZE.
SOE	Source Offset Configuration 0 No source offset configuration error. 1 The last recorded error was a configuration error detected in the TCD.SOFF field. TCD.SOFF is inconsistent with TCD.SSIZE.
DAE	Destination Address Error 0 No destination address configuration error. 1 The last recorded error was a configuration error detected in the TCD.DADDR field. TCD.DADDR is inconsistent with TCD.DSIZE.
DOE	Destination Offset Error 0 No destination offset configuration error. 1 The last recorded error was a configuration error detected in the TCD.DOFF field. TCD.DOFF is inconsistent with TCD.DSIZE.
NCE	NBYTES/CITER Configuration Error 0 No NBYTES/CITER configuration error. 1 The last recorded error was a configuration error detected in the TCD.NBYTES or TCD.CITER fields. TCD.NBYTES is not a multiple of TCD.SSIZE and TCD.DSIZE, or TCD.CITER is equal to zero, or TCD.CITER.E_LINK is not equal to TCD.BITER.E_LINK.
SGE	Scatter/Gather Configuration Error 0 No scatter/gather configuration error. 1 The last recorded error was a configuration error detected in the TCD.DLAST_SGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCD.E_SG is enabled. TCD.DLAST_SGA is not on a 32-byte boundary.
SBE	Source Bus Error 0 No source bus error. 1 The last recorded error was a bus error on a source read.
DBE	Destination Bus Error 0 No destination bus error. 1 The last recorded error was a bus error on a destination write.

9.2.1.3 DMA Enable Request (DMAERQH, DMAERQL)

The DMA Enable Request High (DMAERQH) and DMA Enable Request Low (DMAERQL) registers provide a bit map for the implemented 64 channels to enable the request signal for each channel. DMAERQH supports channels 63–32, while DMAEQRL covers channels 31–00. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the DMASERQ and DMACERQ registers. The DMASERQ and DMACERQ registers are provided so that the request enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the DMAERQH or DMAERQL registers.

The DMA request input signal and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

As a given channel completes the processing of its major iteration count, a flag in the transfer control descriptor (TCD) can affect the ending state of the DMAERQ bit for that channel. If the TCD.D_REQ bit is set, the corresponding DMAERQ bit is cleared, disabling the DMA request. If the D_REQ bit is cleared, the state of the DMAERQ bit is unaffected.

Address: Base + 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ 63	ERQ 62	ERQ 61	ERQ 60	ERQ 59	ERQ 58	ERQ 57	ERQ 56	ERQ 55	ERQ 54	ERQ 53	ERQ 52	ERQ 51	ERQ 50	ERQ 49	ERQ 48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ 47	ERQ 46	ERQ 45	ERQ 44	ERQ 43	ERQ 42	ERQ 41	ERQ 40	ERQ 39	ERQ 38	ERQ 37	ERQ 36	ERQ 35	ERQ 34	ERQ 33	ERQ 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-3. DMA Enable Request Register High (DMAERQH)

Address: Base + 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ 31	ERQ 30	ERQ 29	ERQ 28	ERQ 27	ERQ 26	ERQ 25	ERQ 24	ERQ 23	ERQ 22	ERQ 21	ERQ 20	ERQ 19	ERQ 18	ERQ 17	ERQ 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ 15	ERQ 14	ERQ 13	ERQ 12	ERQ 11	ERQ 10	ERQ 09	ERQ 08	ERQ 07	ERQ 06	ERQ 05	ERQ 04	ERQ 03	ERQ 02	ERQ 01	ERQ 00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-4. DMA Enable Request Register Low (DMAERQL)

Table 9-6. DMAERQH and DMAERQL field descriptions

Field	Description
ERQ*n*	Enable DMA Request *n* 0 The DMA request signal for channel *n* is disabled. 1 The DMA request signal for channel *n* is enabled.

9.2.1.4 DMA Enable Error Interrupt (DMAEEIH, DMAEEIL)

The DMA Enable Error Interrupt High and Low (DMAEEIH and DMAEEIL) registers provide a bit map for the implemented 64 channels to enable the error interrupt signal for each channel. DMAEEIH supports channels 63–32, while DMAEEIL covers channels 31–00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the DMASEEI and DMACEEI registers. The DMASEEI and DMACEEI registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the DMAEEIH or DMAEEIL registers.

The DMA error indicator and this error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted. See Figure 9-5 and Figure 9-6, and Table 9-7 for the DMAEEI definition.

Address: Base + 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EEI 63	EEI 62	EEI 61	EEI 60	EEI 59	EEI 58	EEI 57	EEI 56	EEI 55	EEI 54	EEI 53	EEI 52	EEI 51	EEI 50	EEI 49	EEI 48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EEI 47	EEI 46	EEI 45	EEI 44	EEI 43	EEI 42	EEI 41	EEI 40	EEI 39	EEI 38	EEI 37	EEI 36	EEI 35	EEI 34	EEI 33	EEI 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-5. DMA Enable Error Interrupt High (DMAEEIH) Register

Address: Base + 0x0014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EEI 31	EEI 30	EEI 29	EEI 28	EEI 27	EEI 26	EEI 25	EEI 24	EEI 23	EEI 22	EEI 21	EEI 20	EEI 19	EEI 18	EEI 17	EEI 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EEI 15	EEI 14	EEI 13	EEI 12	EEI 11	EEI 10	EEI 09	EEI 08	EEI 07	EEI 06	EEI 05	EEI 04	EEI 03	EEI 02	EEI 01	EEI 00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-6. DMA Enable Error Interrupt Low (DMAEEIL) Register

Table 9-7. DMAEEIH and DMAEEIL field descriptions

Field	Description
EEI*n*	Enable Error Interrupt *n* 0 The error signal for channel *n* does not generate an error interrupt. 1 The assertion of the error signal for channel *n* generates an error interrupt request.

9.2.1.5 DMA Set Enable Request (DMASERQ)

The DMA Set Enable Request (DMASERQ) register provides a simple memory-mapped mechanism to set a given bit in the DMAERQH and DMAERQL registers to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the DMAERQH or DMAERQL register to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing the entire contents of the DMAERQH and DMAERQL registers to be asserted. Reads of this register return all 0s. See Figure 9-7 and Table 9-8 for the DMASERQ definition.

Address: Base + 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		SERQ[6:0]						
Reset	0	0	0	0	0	0	0	0

Figure 9-7. DMA Set Enable Request Register (DMASERQ)

Table 9-8. DMASERQ field descriptions

Field	Description
SERQ[6:0]	Set Enable Request 0–63 Sets the corresponding bit in the DMAERQH or DMAERQL register. 64–127 Sets all the bits in the DMAERQH and DMAERQL registers.

9.2.1.6 DMA Clear Enable Request (DMACERQ)

The DMA Clear Enable Request (DMACERQ) register provides a simple memory-mapped mechanism to clear a given bit in the DMAERQH and DMAERQL registers to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the DMAERQH or DMAERQL register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAERQH and DMAERQL registers to be cleared, disabling all DMA request inputs. Reads of this register return all 0s. See Figure 9-8 and Table 9-9 for the DMACERQ definition.

Address: Base + 0x0019 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		CERQ[6:0]						
Reset	0	0	0	0	0	0	0	0

Figure 9-8. DMA Clear Enable Request Register (DMACERQ)

Table 9-9. DMACERQ field descriptions

Field	Description
CERQ[6:0]	Clear Enable Request 0–63 Clears the corresponding bit in the DMAERQH or DMAERQL register. 64–127 Clears all the bits in the DMAERQH and DMAERQL registers.

9.2.1.7 DMA Set Enable Error Interrupt (DMASEEI)

The DMASEEI register provides a simple memory-mapped mechanism to set a given bit in the DMAEEIH and DMAEEIL registers to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the DMAEEIH or DMAEEIL register to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing the entire contents of DMAEEIH and DMAEEIL registers to be set to 1s. Reads of this register return all 0s. See Figure 9-9 and Table 9-10 for the DMASEEI definition.

Address: Base + 0x001A Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		SEEI[6:0]						
Reset	0	0	0	0	0	0	0	0

Figure 9-9. DMA Set Enable Error Interrupt Register (DMASEEI)

Table 9-10. DMASEEI field descriptions

Field	Description
SEEI[6:0]	Set Enable Error Interrupt 0–63 Sets the corresponding bit in the DMAEEIH or DMAEEIL register. For example, writing a value of 0b000_1111 (15 d, 0xF) sets the bit for EEI15 in DMAEEIL, allowing the assertion of an error signal on channel 15 to generate an error interrupt request. 64–127 Sets all the bits in the DMAEEIH and DMAEEIL registers.

9.2.1.8 DMA Clear Enable Error Interrupt (DMACEEI)

The DMA Clear Enable Error Interrupt (DMACEEI) register provides a simple memory-mapped mechanism to clear a given bit in the DMAERQH and DMAERQL registers to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the DMAERQH or DMAERQL register to be cleared. A data value of 64 to 127 (regardless of the number of implemented

channels) provides a global clear function, forcing the entire contents of the DMAERQH and DMAERQL to be cleared, disabling all DMA request inputs. Reads of this register return all 0s. See Figure 9-10 and Table 9-11 for the DMACEEI definition.

Address: Base + 0x001B Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		CEEI[6:0]						
Reset	0	0	0	0	0	0	0	0

Figure 9-10. DMA Clear Enable Error Interrupt Register (DMACEEI)

Table 9-11. DMACEEI field descriptions

Field	Description
CEEI[6:0]	Clear Enable Error Interrupt 0–63 Clears the corresponding bit in the DMAERQH or DMAERQL register. 64–127 Clear all the bits in the DMAERQH and DMAERQL registers.

9.2.1.9 DMA Clear Interrupt Request (DMACINT)

The DMA Clear Interrupt Request (DMACINT) register provides a simple memory-mapped mechanism to clear a given bit in the DMAINTH and DMAINTL registers to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the DMAINTH or DMAINTL register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAINTH and DMAINTL registers to be cleared, disabling all DMA interrupt requests. Reads of this register return all 0s. See Figure 9-11 and Table 9-12 for the DMACINT definition.

Address: Base + 0x001C Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		CINT[6:0]						
Reset	0	0	0	0	0	0	0	0

Figure 9-11. DMA Clear Interrupt Request Register (DMACINT)

Table 9-12. DMACINT field descriptions

Field	Description
CINT[6:0]	Clear Interrupt Request 0-63 Clears the corresponding bit in the DMAINTH or DMAINTL register. 64-127 Clears all the bits in the DMAINTH and DMAINTL registers.

9.2.1.10 DMA Clear Error (DMACERR)

The DMA Clear Error (DMACERR) register provides a simple memory-mapped mechanism to clear a given bit in the DMAERRH and DMAERRL registers to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the DMAERRH or DMAERRL register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAERRH and DMAERRL registers to be cleared, clearing all channel error indicators. Reads of this register return all 0s. See Figure 9-12 and Table 9-13 for the DMACERR definition.

Address: Base + 0x001D Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		CERR[6:0]						
Reset	0	0	0	0	0	0	0	0

Figure 9-12. DMA Clear Error Register (DMACERR)

Table 9-13. DMACERR field descriptions

Field	Description
CERR[6:0]	Clear Error Indicator 0–63 Clears the corresponding bit in the DMAERRH or DMAERRL register. 64–127 Clears all the bits in the DMAERRH and DMAERRL registers.

9.2.1.11 DMA Set START Bit (DMASSRT)

The DMA Set START Bit (DMASSRT) register provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding Transfer Control Descriptor to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing all START bits to be set. Reads of this register return all 0s. See Figure 9-13 and Table 9-14 for the TCD.START bit definition.

Address: Base + 0x001E Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		SSRT[6:0]						
Reset	0	0	0	0	0	0	0	0

Figure 9-13. DMA Set START Bit Register (DMASSRT)

Table 9-14. DMASSRT field descriptions

Field	Description
SSRT[6:0]	Set START Bit (Channel Service Request) 0–63 Sets the corresponding channel's TCD.START bit. 64–127 Sets all TCD.START bits.

9.2.1.12 DMA Clear DONE Status (DMACDNE)

The DMA Clear DONE Status (DMACDNE) register provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding Transfer Control Descriptor to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing all DONE bits to be cleared. Reads of this register return all 0s. See Figure 9-14 and Table 9-15 for the TCD.DONE bit definition.

Address: Base + 0x001F Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		CDNE[6:0]						
Reset	0	0	0	0	0	0	0	0

Figure 9-14. DMA Clear DONE Status Register (DMACDNE)

Table 9-15. DMACDNE field descriptions

Field	Description
CDNE[6:0]	Clear DONE Status Bit 0–63 Clears the corresponding channel's DONE bit. 64–127 Clears all the TCD.DONE bits.

9.2.1.13 DMA Interrupt Request (DMAINTH, DMAINTL)

The DMA Interrupt Request High (DMAINTH) and DMA Interrupt Request Low (DMAINTL) registers provide a bit map for the implemented 64 channels signaling the presence of an interrupt request for each channel. DMAINTH supports channels 63–32, while DMAINTL covers channels 31–00. The DMA_ENGINE signals the occurrence of a programmed interrupt upon the completion of a data transfer as defined in the transfer control descriptor by setting the appropriate bit in this register. The outputs of this register are directly routed to the platform's interrupt controller. During the execution of the interrupt service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the DMACINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the DMACINT register. On writes to the DMAINT, a 1 in any bit position clears the corresponding channel's interrupt request. A 0 in any bit position has no effect on the corresponding channel's current interrupt status. The DMACINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the DMAINTH or DMAINTL registers.

See Figure 9-15 and Figure 9-16, and Table 9-16 for the DMAINT definition.

Address: Base + 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT
W	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT
W	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-15. DMA Interrupt Request Register High (DMAINTH)

Address: Base + 0x0024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT
W	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT	INT
W	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-16. DMA Interrupt Request Register Low (DMAINTL)

Table 9-16. DMAINTH and DMAINTL field descriptions

Field	Description
INT*n*	DMA Interrupt Request *n* 0 The interrupt request for channel *n* is cleared. 1 The interrupt request for channel *n* is active.

9.2.1.14 DMA Error (DMAERRH, DMAERRL)

The DMA Error High (DMAERRH) and DMA Error Low (DMAERRL) registers provide a bit map for the implemented 64 channels signaling the presence of an error for each channel. DMAERRH supports channels 63–32, while DMAERRL covers channels 31–00. The DMA_ENGINE signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the DMAEEI register, then logically summed across groups of 16, 32, and 64 channels to form several group error interrupt requests routed to the platform's interrupt controller. During the execution of the interrupt service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error interrupt request. Typically, a write to the DMACERR register in the interrupt service routine is used for this purpose. Recall the normal DMA channel

completion indicators, setting the transfer control descriptor done flag and the possible assertion of an interrupt request, are not affected when an error is detected.

The contents of this register can also be polled and a non-zero value indicates the presence of a channel error, regardless of the state of the DMAEEI register. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the DMACERR register. On writes to the DMAERR, a 1 in any bit position clears the corresponding channel's error status. A 0 in any bit position has no effect on the corresponding channel's current error status. The DMACERR register is provided so the error indicator for a single channel can easily be cleared. See Figure 9-17 and Figure 9-18, and Table 9-17 for the DMAERR definition.

Address: Base + 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR 63	ERR 62	ERR 61	ERR 60	ERR 59	ERR 58	ERR 57	ERR 56	ERR 55	ERR 54	ERR 53	ERR 52	ERR 51	ERR 50	ERR 49	ERR 48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR 47	ERR 46	ERR 45	ERR 44	ERR 43	ERR 42	ERR 41	ERR 40	ERR 39	ERR 38	ERR 37	ERR 36	ERR 35	ERR 34	ERR 33	ERR 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-17. DMA Error Register High (DMAERRH)

Address: Base + 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR 31	ERR 30	ERR 29	ERR 28	ERR 27	ERR 26	ERR 25	ERR 24	ERR 23	ERR 22	ERR 21	ERR 20	ERR 19	ERR 18	ERR 17	ERR 16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR 15	ERR 14	ERR 13	ERR 12	ERR 11	ERR 10	ERR 09	ERR 08	ERR 07	ERR 06	ERR 05	ERR 04	ERR 03	ERR 02	ERR 01	ERR 00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-18. DMA Error Register Low (DMAERRL)

Table 9-17. DMAERRH and DMAERRL field descriptions

Field	Description
ERR*n*	DMA Error *n* 0 An error in channel *n* has not occurred. 1 An error in channel *n* has occurred.

9.2.1.15 DMA Hardware Request Status (DMAHRSH, DMAHRSL)

The DMA Hardware Request Status High (DMAHRSH) and DMA Hardware Request Status Low (DMAHRSL) registers provide a bit map for the implemented 64 channels' current hardware request status. DMAHRSH supports channels 63–32, while DMAHRSL covers channels 31–00. The hardware request status reflects the current state of the registered and qualified (via the DMAERQ field) request lines as seen by DMA arbitration logic. This view into the hardware request signals may be used for debug purposes. See Figure 9-19 and Figure 9-20, and Table 9-18 for the DMAHRS definition.

Address: Base + 0x0030 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HRS 63	HRS 62	HRS 61	HRS 60	HRS 59	HRS 58	HRS 57	HRS 56	HRS 55	HRS 54	HRS 53	HRS 52	HRS 51	HRS 50	HRS 49	HRS 48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HRS 47	HRS 46	HRS 45	HRS 44	HRS 43	HRS 42	HRS 41	HRS 40	HRS 39	HRS 38	HRS 37	HRS 36	HRS 35	HRS 34	HRS 33	HRS 32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-19. DMA Hardware Request Status Register High (DMAHRSH)

Address: Base + 0x0034 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HRS 31	HRS 30	HRS 29	HRS 28	HRS 27	HRS 26	HRS 25	HRS 24	HRS 23	HRS 22	HRS 21	HRS 20	HRS 19	HRS 18	HRS 17	HRS 16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HRS 15	HRS 14	HRS 13	HRS 12	HRS 11	HRS 10	HRS 09	HRS 08	HRS 07	HRS 06	HRS 05	HRS 04	HRS 03	HRS 02	HRS 01	HRS 00
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-20. DMA Hardware Request Status Register Low (DMHRSL)

Table 9-18. DMAHRSH and DMAHRSL field descriptions

Field	Description
HSA*n*	DMA Hardware Request Status 0 A Hardware service request for channel *n* is not present. 1 A Hardware service request for channel *n* is present. **Note:** The hardware request status reflects the state of the request as seen by the arbitration logic. Therefore, this status is affected by the DMAERQ*n* bit.

9.2.1.16 DMA Channel *n* Priority (DCHPRI*n*), *n* = [0,..., {15, 31, 63}

When the fixed-priority channel arbitration mode is enabled (DMACR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value (0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc.). Software must program the channel priorities with unique values; otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15. When read, the GRPPRI bits of the DCHPRIn register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRI*n* registers. The group priority is assigned in the DMACR. See Figure 9-1 and Table 9-4 for the DMACR definition.

Channel preemption is enabled on a per channel basis by setting the ECP bit in the DCHPRIn register. Channel preemption allows the executing channel's data transfers to be temporarily suspended in favor of starting a higher priority channel. After the preempting channel has completed all of its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption (attempting to preempt a preempting channel) is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is only available when fixed arbitration is selected for group and channel arbitration modes. See Figure 9-21 and Table 9-19 for the DCHPRI*n* definition.

Address: Base + 0x0100 + *n* Access: User read/write

	0	1	2	3	4	5	6	7
R	ECP	0	GRPPRI[1:0]		CHPRI[3:0]			
W								
Reset	0	0	—[1]	—[1]	—[2]	—[2]	—[2]	—[2]

Figure 9-21. DMA Channel *n* Priority Register (DCHPRI*n*)

1 GRPPRI[1:0] equals the corresponding group's GRP*x*PRI value of the DMACR*n* register.

2 CHPRI[3:0] defaults to the values for channel number (*n*) after reset.

Table 9-19. DCHPRI*n* field descriptions

Field	Description
ECP	Enable Channel Preemption 0 Channel *n* cannot be suspended by a higher priority channel's service request. 1 Channel *n* can be temporarily suspended by the service request of a higher priority channel.
GRPPRI[1:0]	Channel *n* Current Group Priority. Group priority assigned to this channel group when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored.
CHPRI[3:0]	Channel *n* Arbitration Priority. Channel priority when fixed-priority arbitration is enabled.

9.2.1.17 Transfer Control Descriptor (TCD)

Each channel requires a 32-byte transfer control descriptor (TCD) for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order. The definitions of the TCD are presented as eight 32-bit values. Table 9-20 is a 32-bit view of the basic TCD structure.

Table 9-20. TCD*n* 32-bit Memory Structure

TCD*n* Word	DMA Offset	TCD*n* Field	
0	0x1000 + (32 × *n*) + 0x0000	Source Address (SADDR)	
1	0x1000 + (32 × *n*) + 0x0004	Transfer Attributes (SMOD, SSIZE, DMOD, DSIZE)	Signed Source Address Offset (SOFF)
2	0x1000 + (32 × *n*) + 0x0008	Inner Minor Byte Count (NBYTES)	
3	0x1000 + (32 × *n*) + 0x000C	Last Source Address Adjustment (SLAST)	
4	0x1000 + (32 × *n*) + 0x0010	Destination Address (DADDR)	
5	0x1000 + (32 × *n*) + 0x0014	Current Major Iteration Count (CITER)	Signed Destination Address Offset (DOFF)
6	0x1000 + (32 × *n*) + 0x0018	Last Destination Address Adjustment/Scatter Gather Address (DLAST_SGA)	
7	0x1000 + (32 × *n*) + 0x001C	Beginning Major Iteration Count (BITER)	Channel Control/Status (BWC, MAJOR.LINKCH, DONE, ACTIVE, MAJOR.E_LINK, E_SG, D_REQ, INT_HALF, INT_MAJ, START)

Table 9-21 shows the addresses of the eight TCD words in TCD00. The other TCD channels repeat this addressing, based on their individual addresses as shown in Table 9-1.

Table 9-21. Sample TCD Memory Structure for TCD00

TCD00 Word	Offset from DMA_BASE
0	0x1000
1	0x1004
2	0x1008
3	0x100C
4	0x1010
5	0x1014
6	0x1018
7	0x101C

Figure 9-22 and Table 9-22 define word 0 of the TCD*n* structure, the SADDR field.

Address: TCDn Base + 0x1000 + (32 × *n*) + 0x0000
See Table 9-1 and Table 9-21.

Access: User read/write

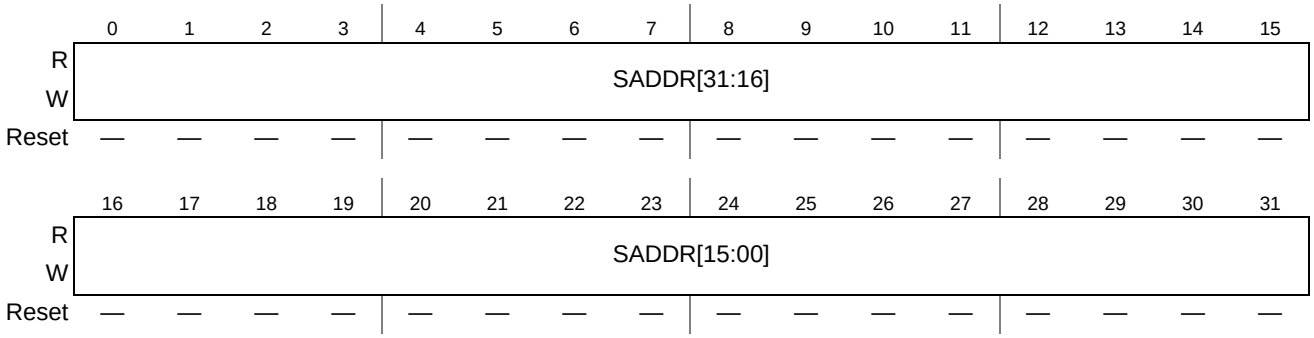


Figure 9-22. TCD*n* Word 0 (TCD*n*.SADDR) Field

Table 9-22. TCD*n* Word 0 field descriptions

Field	Description
SADDR[31:0]	Source Address. Memory address pointing to the source data.

Figure 9-23 and Table 9-23 define word 1 of the TCD*n* structure, the SOFF and transfer attribute fields.

Address: Base + 0x1000 + (32 × *n*) + 0x0004
See Table 9-1 and Table 9-21.

Access: User read/write

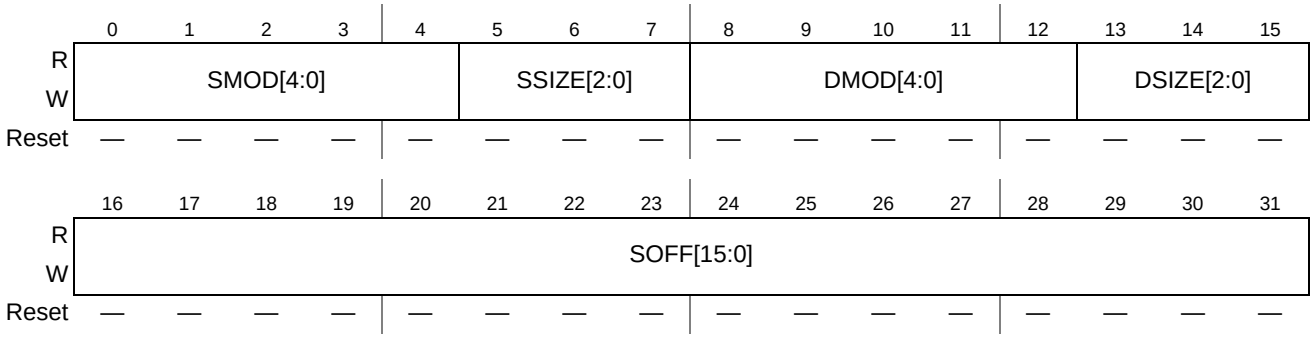


Figure 9-23. TCD*n* Word 1 (TCD*n*.{SOFF,SMOD,SSIZE,DMOD,DSIZE}) Field

Table 9-23. TCD*n* Word 1 field descriptions

Field	Description
SMOD[4:0]	Source address modulo 0 Source address modulo feature is disabled. non-0 The value defines a specific address bit selected to be the value after SADDR + SOFF calculation is performed on the original register value. This feature provides the ability to easily implement a circular data queue. For data queues requiring power-of-two size bytes, the queue should be based at a zero-modulo-size address and the smod field set to the appropriate value to freeze the upper address bits. The bit select is defined as ((1 << smod[4:0]) – 1) where a resulting 1 in a bit location selects the next state address for the corresponding address bit location and a 0 selects the original register value for the corresponding address bit location. For this application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a zero-modulo-size range.

Table 9-23. TCD*n* Word 1 field descriptions (Continued)

Field	Description
SSIZE[2:0]	Source data transfer size 000 8-bit 001 16-bit 010 32-bit 011 Reserved 100 16-byte 101 32-byte 110 Reserved 111 Reserved The attempted specification of a reserved source size produces a configuration error.
DMOD[4:0]	Destination address modulo. See the SMOD[5:0] definition.
DSIZE[2:0]	Destination data transfer size. See the SSIZE[2:0] definition.
SOFF[15:0]	Source address signed offset. Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

Figure 9-24 and Table 9-24 define word 2 of the TCD*n* structure, the NBYTES field.

Address: Base + 0x1000 + (32 × *n*) + 0x0008
See Table 9-1 and Table 9-21.

Access: User read/write

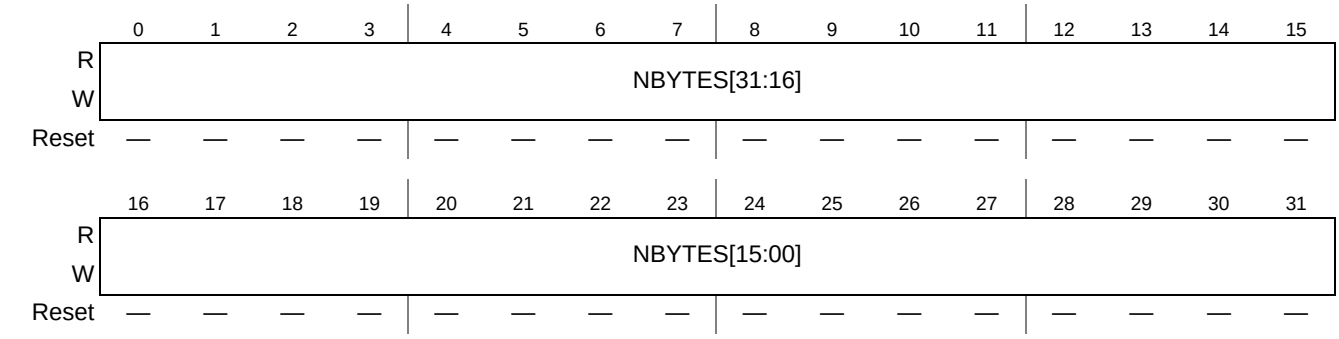


Figure 9-24. TCD*n* Word 2 (TCD*n*.NBYTES) Field

Table 9-24. TCD*n* Word 2 field descriptions

Field	Description
NBYTES[31:0]	Inner Minor Byte Transfer Count. Number of bytes to be transferred in each service request of the channel. As a channel is activated, the contents of the appropriate TCD is loaded into the DMA_ENGINE, and the appropriate reads and writes perform until the complete byte transfer count has been transferred. This is an indivisible operation and cannot be stalled or halted. After the minor count is exhausted, the current values of the SADDR and DADDR are written back into the local memory, the major iteration count is decremented and restored to the local memory. If the major iteration count is completed, additional processing is performed. The NBYTES value of 0x0000_0000 is interpreted as 0x1_0000_0000, thus specifying a 4 GB transfer.

Figure 9-25 and Table 9-25 define word 3 of the TCD*n* structure, the SLAST field.

Address: Base + 0x1000 + (32 × *n*) + 0x000C
See Table 9-1 and Table 9-21.

Access: User read/write

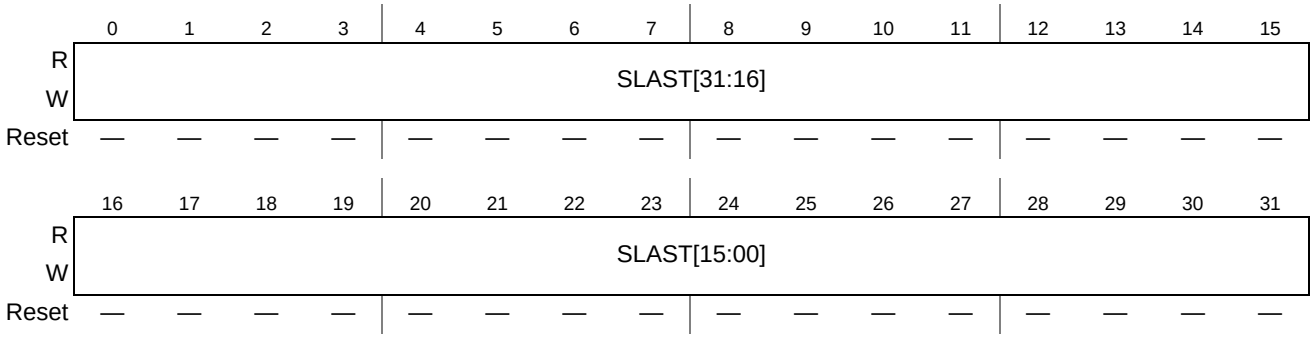


Figure 9-25. TCDn Word 3 (TCDn.slast) Field

Table 9-25. TCDn Word 3 (TCDn.slast) field descriptions

Field	Description
SLAST[31:0]	Last source address adjustment. Adjustment value added to the source address at the completion of the outer major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure.

Figure 9-26 and Table 9-26 define word 4 of the TCD*n* structure, the DADDR field.

Address: Base + 0x1000 + (32 × *n*) + 0x0010
See Table 9-1 and Table 9-21.

Access: User read/write

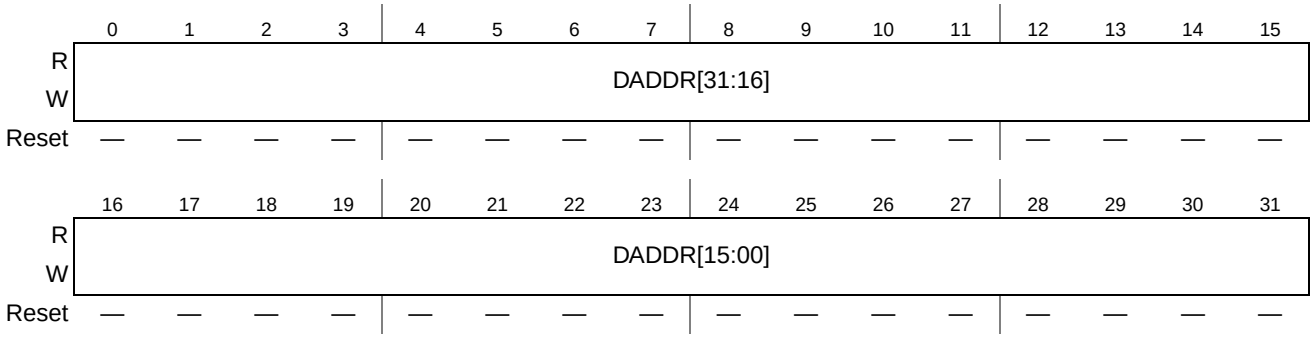


Figure 9-26. TCD*n* Word 4 (TCD*n*.DADDR) Field

Table 9-26. TCD*n* Word 4 field descriptions

Field	Description
DADDR[31:0]	Destination address. Memory address pointing to the destination data.

Figure 9-27 and Table 9-27 define word 5 of the TCD*n* structure, the CITER and DOFF fields.

Address: Base + 0x1000 + (32 × *n*) + 0x14
See Table 9-1 and Table 9-21.

Access: User read/write

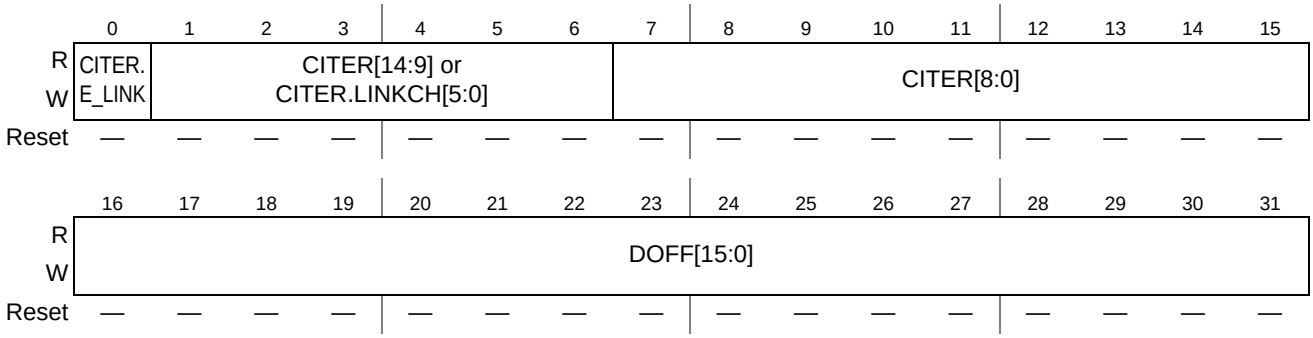


Figure 9-27. TCD*n* Word 5 (TCD*n*.{CITER,DOFF}) Field

Table 9-27. TCD*n* Word 5 field descriptions

Field	Description
CITER.E_LINK	Enable Channel-to-Channel Linking on Minor Loop Complete. As the channel completes the inner minor loop, this flag enables the linking to another channel, defined by CITER.LINKCH[5:0]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD.START bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJOR.E_LINK channel linking. This bit must be equal to the BITER.E_LINK bit otherwise a configuration error is reported. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
CITER[14:9] or CITER.LINKCH[5:0]	Current Major Iteration Count or Link Channel Number If TCD.CITER.E_LINK equals 0, No channel-to-channel linking (or chaining) is performed after the inner minor loop is exhausted. TCD word 5, bits [30:25] are used to form a 15-bit CITER field. or After the minor loop is exhausted, the DMA_ENGINE initiates a channel service request at the channel defined by CITER.LINKCH[5:0] by setting that channel's TCD.START bit. The value contained in CITER.LINKCH[5:0] must not exceed the number of implemented channels.
CITER[8:0]	Current major iteration count.
DOFF[15:0]	Destination address signed offset.

Figure 9-28 and Table 9-28 define word 6 of the TCD*n* structure, the DLAST_SGA field.

Address: Base + 0x1000 + (32 × *n*) + 0x0018
See Table 9-1 and Table 9-21.
Access: User read/write

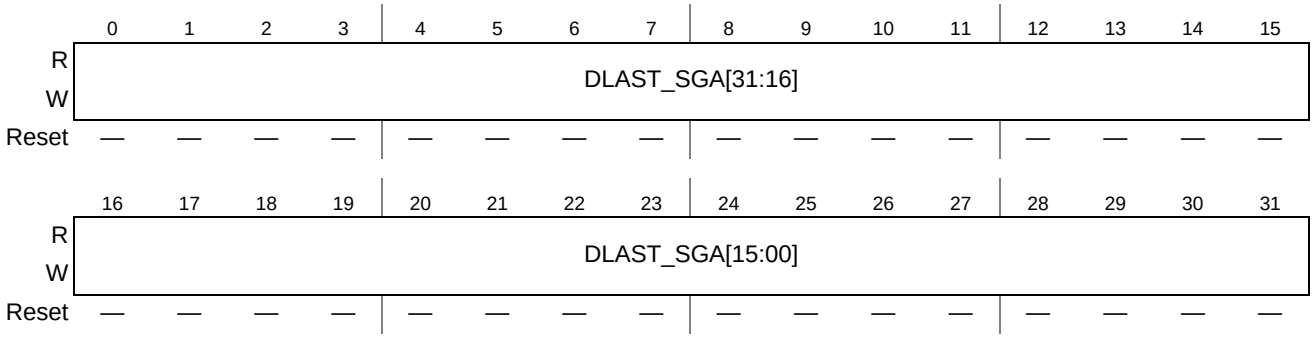


Figure 9-28. TCD*n* Word 6 (TCDn.DLAST_SGA)

Table 9-28. TCD*n* Word 6 field descriptions

Field	Description
DLAST_SGA[31:0]	Last destination address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather) If TCD.e_sg equals 0, Adjustment value is added to the destination address at the completion of the outer major iteration count. This value can be applied to restore the destination address to the initial value or adjust the address to reference the next data structure. or This address points to the beginning of a 0-modulo-32 region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32 or a configuration error is reported.

Figure 9-29 and Table 9-29 define word 7 of the TCD*n* structure, the BITER and control/status fields.

Address: Base + 0x1000 + (32 × n) + 0x001C
See Table 9-1 and Table 9-21.
Access: User read/write

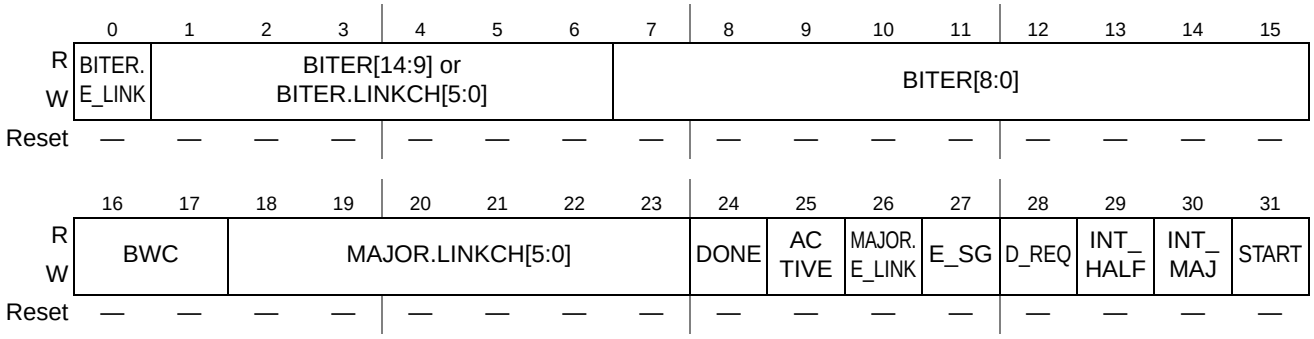


Figure 9-29. TCD*n* Word 7 (TCD*n*.{BITER,control/status}) Fields

Table 9-29. TCD*n* Word 7 field descriptions

Field	Description
BITER.E_LINK	Enable channel-to-channel linking on major loop complete. This is the initial value copied into the CITER.E_LINK field when the major loop is completed. The CITER.E_LINK field controls channel linking during channel execution. This bit must be equal to the CITER.E_LINK bit otherwise a configuration error is reported. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
BITER[14:9] or BITER.LINKCH [5:0]	Beginning major iteration count or beginning link channel number. This is the initial value copied into the CITER field or CITER.LINKCH field when the major loop is completed. The CITER fields controls the interation count and linking during channel execution. If TCD.BITER.E_LINK equals 0, No channel-to-channel linking (or chaining) is performed after the inner minor loop is exhausted. TCD word 5, bits [30:25] are used to form a 15-bit BITER field. or After the minor loop is exhausted, the DMA_ENGINE initiates a channel service request at the channel defined by BITER.LINKCH[5:0] by setting that channel's TCD.START bit. The value contained in BITER.LINKCH[5:0] must not exceed the number of implemented channels.
BITER[8:0]	Beginning major iteration count. This is the initial value copied into the CITER field or ciTer.LINKCH field when the major loop is completed. The CITER fields controls the interation count and linking during channel execution. This 9- or 15-bit count represents the beginning major loop count for the channel. As the major iteration count is exhausted, the contents of the entire 16-bit BITER entry is reloaded into the 16-bit CITER entry. When the BITER field is initially loaded by software, it must be set to the same value as that contained in the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.
BWC[1:0]	Bandwidth control. This two-bit field provides a mechanism to effectively throttle the amount of bus bandwidth consumed by the DMA. In general, as the DMA processes the inner minor loop, it continuously generates read/write, read/write, etc. sequences until the minor count is exhausted. This field forces the DMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the platform's cross-bar arbitration switch. To minimize start-up latency, bandwidth control stalls are suppressed for the first two AHB bus cycles and after the last write of each minor loop. The dynamic priority elevation setting elevates the priority of the DMA as seen by the cross-bar arbitration switch for the executing channel. Dynamic priority elevation is suppressed during the first two AHB bus cycles. 00 No DMA_ENGINE stalls. 01 Dynamic priority elevation. 10 DMA_ENGINE stalls for four cycles after each r/w. 11 DMA_ENGINE stalls for eight cycles after each r/w.
MAJOR.LINKCH[5:0]	Link channel number If TCD.MAJOR.E_LINK equals 0, no channel-to-channel linking (or chaining) is performed after the outer major loop counter is exhausted. or After the major loop counter is exhausted, the DMA_ENGINE initiates a channel service request at the channel defined by MAJOR.LINKCH[5:0] by setting that channel's TCD.START bit. The value contained in MAJOR.LINKCH[5:0] must not exceed the number of implemented channels.

Table 9-29. TCD*n* Word 7 field descriptions (Continued)

Field	Description
DONE	Channel done. This flag indicates the DMA has completed the outer major loop. It is set by the DMA_ENGINE as the CITER count reaches 0; it is cleared by software or the hardware when the channel is activated. This bit must be cleared to write the MAJOR.E_LINK or E_SG bits.
ACTIVE	Channel active. This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the DMA_ENGINE as the inner minor loop completes or if any error condition is detected.
MAJOR.E_LINK	Enable channel-to-channel linking on major loop complete. As the channel completes the outer major loop, this flag enables the linking to another channel, defined by MAJOR.LINKCH[5:0]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD.START bit of the specified channel. To support the dynamic linking coherency model, this field is forced to zero when written to while the TCD.DONE bit is set. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
E_SG	Enable scatter/gather processing. As the channel completes the outer major loop, this flag enables scatter/gather processing in the current channel. If enabled, the DMA_ENGINE uses DLAST_SGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure that is loaded as the transfer control descriptor into the local memory. To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCD.DONE bit is set. 0 The current channel's TCD is normal format. 1 The current channel's TCD specifies a scatter gather format. The DLAST_SGA field provides a memory pointer to the next TCD to be loaded into this channel after the outer major loop completes its execution.
D_REQ	Disable request. If this flag is set, the DMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero. 0 The channel's DMAERQ bit is not affected. 1 The channel's DMAERQ bit is cleared when the outer major loop is complete.
INT_HALF	Enable an interrupt when major counter is half complete. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the DMAINT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the DMA_ENGINE is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered schemes or other types of data movement where the processor needs an early indication of the transfer's progress. The halfway complete interrupt is disabled when BITER values are less than 2. 0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled.
INT_MAJ	Enable an interrupt when major iteration count completes. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the DMAINT register when the current major iteration count reaches 0. 0 The end-of-major loop interrupt is disabled. 1 The end-of-major loop interrupt is enabled.
START	Channel start. If this flag is set, the channel is requesting service. The DMA hardware automatically clears this flag after the channel begins execution. 0 The channel is not explicitly started. 1 The channel is explicitly started via a software initiated service request.

9.3 Initialization/Application Information

9.3.1 DMA Initialization

A typical initialization of the DMA would be:

1. Write the DMACR register if a configuration other than the default is desired.
2. Write the channel priority levels into the DCHPRIn registers if a configuration other than the default is desired.
3. Enable error interrupts in the DMAEEI registers if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the DMAERQ register.
6. Request channel service by either software (setting the TCD.START bit) or by hardware (slave device asserting its IPD_REQ signal).

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The DMA_ENGINE reads the entire TCD for the selected channel into its internal address path module. As the TCD is being read, the first transfer is initiated on the AHB bus unless a configuration error is detected. Transfers from the source (as defined by the source address, TCD.SADDR) to the destination (as defined by the destination address, TCD.DADDR) continue until the specified number of bytes (TCD.NBYTES) have been transferred. When the transfer is complete, the DMA_ENGINE'S local TCD.SADDR, TCD.DADDR, and TCD.CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing is executed, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

9.3.2 DMA Programming Errors

The DMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per-channel basis, with the exception of two errors, group priority error (GPE) and channel priority error (CPE) in the DMAES register.

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the DMAES register. If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

The typical application enables error interrupts for all channels. You receive an error interrupt, but the channel number for the DMAERR register and the error interrupt request line may be wrong because they reflect the selected channel.

Channel priority errors are identified within a group after that group has been selected as the active group. For example:

1. The DMA is configured for fixed-group and fixed-channel arbitration modes.
2. Group3 is the highest priority and all channels are unique in that group.
3. Group2 is the next highest priority and has two channels with the same priority level.
4. If Group3 has any service requests, those requests are executed.

5. After all of Group3 requests have completed, Group2 is the next active group.
6. If Group2 has a service request, an undefined channel in Group2 is selected and a channel priority error occurs.
7. This repeats until the all of the Group2 requests have been removed or a higher priority Group3 request comes in.

A group priority error is global and any request in any group causes a group priority error.

In general, if priority levels are not unique, the highest (channel/group) priority with an active request is selected, but the lowest numbered (channel/group) with that priority is selected by arbitration and executed by the DMA_ENGINE. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

9.3.3 DMA Arbitration Mode Considerations

9.3.3.1 Fixed Group Arbitration, Fixed Channel Arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group is selected to execute. If the DMA is programmed so the channels within one group use fixed priorities and that group is assigned the highest fixed priority of all groups, that group may take all the bandwidth of the DMA controller. No other groups are serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request.

The advantage of this scenario is that latency can be small for channels that need to be serviced quickly.

Preemption is available in this scenario only.

9.3.3.2 Round Robin Group Arbitration, Fixed Channel Arbitration

The occurrence of one or more DMA requests from one or more groups, the channel with the highest priority from a specific group is serviced first. Groups are serviced starting with the highest group number with a service request and rotating through to the lowest group number containing a service request.

After the channel request is serviced, the group round robin algorithm selects the highest pending request from the next group in the round robin sequence. Servicing continues round robin, always servicing the highest priority channel in the next group in the sequence or skipping a group if it has no pending requests.

If a channel requests service at a rate that equals or exceeds the round robin service rate, that channel is always serviced before lower priority channels in the same group. Therefore, the lower priority channels are never serviced.

The advantage of this scenario is that no one group consumes all the DMA bandwidth.

The highest priority channel selection latency is potentially greater than fixed/fixed arbitration.

Excessive request rates on high priority channels could prevent the servicing of lower priority channels in the same group.

9.3.3.3 Round Robin Group Arbitration, Round Robin Channel Arbitration

Groups are serviced as described in section Section 9.3.3.2, “Round Robin Group Arbitration, Fixed Channel Arbitration,” but channels are serviced in channel number order this time. Only one channel is serviced from each requesting group for each round robin pass through the groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to channel priority levels.

Because channels are serviced in round robin manner, any channel that generates DMA requests faster than a combination of the group round robin service rate and the channel service rate for its group does not prevent the servicing of other channels in its group. Any DMA requests not serviced are simply lost, but at least one channel is serviced.

This scenario ensures all channels are guaranteed service at some point, regardless of the request rates. However, the potential latency could be quite high.

All channels are treated equally. Priority levels are not used in round robin mode.

9.3.3.4 Fixed Group Arbitration, Round Robin Channel Arbitration

The highest priority group with a request is serviced. Lower priority groups are serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in section Section 9.3.3.1, “Fixed Group Arbitration, Fixed Channel Arbitration,” but all the channels in the highest priority group are serviced.

Service latency is short on the highest priority group, but could potentially become longer as the group priority decreases.

9.3.4 DMA Transfer

9.3.4.1 Single Request

To perform a single transfer of *n* bytes of data with one activation, set the major loop to one (TCD.CITER = TCD.BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the TCD.DONE bit is set and an interrupt is generated if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The DMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a word wide port located at 0x2000. The address offsets are programmed in increments to match the size of the transfer; one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCD.citer =TCD.biter = 1
```

```
TCD.nbytes=16
TCD.saddr =0x1000
TCD.soff  =1
TCD.ssize =0
TCD.slast =-16
TCD.daddr =0x2000
TCD.doff  =4
TCD.dsize =2
TCD.dlast_sga=-16
TCD.int_maj =1
TCD.start =1       (TCD.word7 should be written last after all other fields have
                                                                  been initialized)
All other TCD fields = 0
```

These settings generate the following sequence of events:

1. IPS write to the TCD.START bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. DMA_ENGINE writes: TCD.DONE = 0, TCD.START = 0, TCD.ACTIVE = 1.
4. DMA_ENGINE reads: channel TCD data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a) READ_BYTE(0x1000), READ_BYTE(0x1001), READ_BYTE(0x1002), READ_BYTE(0x1003).
   b) WRITE_WORD(0x2000) → first iteration of the minor loop.
   c) READ_BYTE(0x1004), READ_BYTE(0x1005), READ_BYTE(0x1006), READ_BYTE(0x1007).
   d) WRITE_WORD(0x2004) → second iteration of the minor loop.
   e) READ_BYTE(0x1008), READ_BYTE(0x1009), READ_BYTE(0x100A), READ_BYTE(0x100B).
   f) WRITE_WORD(0x2008) → third iteration of the minor loop
   g) READ_BYTE(0x100C), READ_BYTE(0x100D), READ_BYTE(0x100E), READ_BYTE(0x100F).
   h) WRITE_WORD(0x200C) → last iteration of the minor loop → major loop complete.
6. DMA_ENGINE writes: TCD.SADDR = 0x1000, TCD.DADDR = 0x2000, TCD.CITER = 1 (TCD.BITER).
7. DMA_ENGINE writes: TCD.ACTIVE = 0, TCD.DONE = 1, DMAINT[*n*] = 1.
8. The channel retires.

The DMA becomes idle or services the next channel.

9.3.4.2 Multiple Requests

The next example is the same as previous with the exception of transferring 32 bytes via two hardware requests. The only fields that change are the major loop iteration count and the final address offsets. The DMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the DMAERQ register, channel service requests are initiated by the slave device.

```
TCD.citer =  TCD.biter  =  2
TCD.slast = -32
TCD.dlast_sga= -32
```

These settings generate the following sequence of events:

1. First hardware (IPD_REQ) request for channel service.
2. The channel is selected by arbitration for servicing.
3. DMA_ENGINE writes: TCD.done = 0, TCD.start = 0, TCD.active = 1.
4. DMA_ENGINE reads: channel TCD data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a) READ_BYTE(0x1000), READ_BYTE(0x1001), READ_BYTE(0x1002), READ_BYTE(0x1003).
   b) WRITE_WORD(0x2000) → first iteration of the minor loop.
   c) READ_BYTE(0x1004), READ_BYTE(0x1005), READ_BYTE(0x1006), READ_BYTE(0x1007).
   d) WRITE_WORD(0x2004) → second iteration of the minor loop.
   e) READ_BYTE(0x1008), READ_BYTE(0x1009), READ_BYTE(0x100A), READ_BYTE(0x100B).
   f) WRITE_WORD(0x2008) → third iteration of the minor loop.
   g) READ_BYTE(0x100C), READ_BYTE(0x100D), READ_BYTE(0x100E), READ_BYTE(0x100F).
   h) WRITE_WORD(0x200C) → last iteration of the minor loop.
6. DMA_ENGINE writes: TCD.SADDR = 0x1010, TCD.DADDR = 0x2010, TCD.CITER = 1.
7. DMA_ENGINE writes: TCD.ACTIVE = 0.
8. The channel retires → one iteration of the major loop.

The DMA becomes idle or services the next channel.

9. Second hardware (IPD_REQ) requests channel service.
10. The channel is selected by arbitration for servicing.
11. DMA_ENGINE writes: TCD.DONE = 0, TCD.START = 0, TCD.ACTIVE = 1.
12. DMA_ENGINE reads: channel TCD data from local memory to internal register file,
13. The source to destination transfers are executed as follows:
    a) READ_BYTE(0x1010), READ_BYTE(0x1011), READ_BYTE(0x1012), READ_BYTE(0x1013).
    b) WRITE_WORD(0x2010) → first iteration of the minor loop.
    c) READ_BYTE(0x1014), READ_BYTE(0x1015), READ_BYTE(0x1016), READ_BYTE(0x1017).
    d) WRITE_WORD(0x2014) → second iteration of the minor loop.
    e) READ_BYTE(0x1018), READ_BYTE(0x1019), READ_BYTE(0x101A), READ_BYTE(0x101B).
    f) WRITE_WORD(0x2018) → third iteration of the minor loop.

g) READ_BYTE(0x101C), READ_BYTE(0x101D), READ_BYTE(0x101E), READ_BYTE(0x101F).

h) WRITE_WORD(0x201C) → last iteration of the minor loop → major loop complete.

14. DMA_ENGINE writes: TCD.SADDR = 0x1000, TCD.DADDR = 0x2000, TCD.CITER = 2 (TCD.BITER).
15. DMA_ENGINE writes: TCD.ACTIVE = 0, TCD.DONE = 1, DMAINT[*n*] = 1.
16. The channel retires → major loop complete.

The DMA becomes idle or services the next channel.

9.3.5 TCD Status

9.3.5.1 Minor Loop Complete

There are two methods to test for minor loop completion when using software initiated service requests. The first method is to read the TCD.CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD.START bit and the TCD.ACTIVE bit. The minor loop complete condition is indicated by both bits reading zero after the TCD.START was written to a one. Polling the TCD.ACTIVE bit may be inconclusive because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

1. TCD.START = 1, TCD.ACTIVE = 0, TCD.DONE = 0 (channel service request via software).
2. TCD.START = 0, TCD.ACTIVE = 1, TCD.DONE = 0 (channel is executing).
3. TCD.START = 0, TCD.ACTIVE = 0, TCD.DONE = 0 (channel completed minor loop and is idle).
4. TCD.START = 0, TCD.ACTIVE = 0, TCD.DONE = 1 (channel completed major loop and is idle).

The best method to test for minor loop completion when using hardware initiated service requests is to read the TCD.CITER field and test for a change. The hardware request and acknowledge handshakes signals are not visible in the programmer’s model.

The TCD status bits execute the following sequence for a hardware activated channel:

1. IPD_REQ asserts (channel service request via hardware).
2. TCD.START = 0, TCD.ACTIVE = 1, TCD.DONE = 0 (channel is executing).
3. TCD.START = 0, TCD.ACTIVE = 0, TCD.DONE = 0 (channel completed minor loop and is idle).
4. TCD.START = 0, TCD.ACTIVE = 0, TCD.DONE = 1 (channel completed major loop and is idle).

For both activation types, the major loop complete status is explicitly indicated via the TCD.DONE bit.

The TCD.START bit is cleared automatically when the channel begins execution regardless of how the channel was activated.

9.3.5.2 Active Channel TCD Reads

The DMA reads back the true TCD.SADDR, TCD.DADDR, and TCD.NBYTES values if read while a channel is executing. The true values of the SADDR, DADDR, and NBYTES are the values the DMA_ENGINE is currently using in its internal register file and not the values in the TCD local memory for that channel. The addresses (SADDR and DADDR) and NBYTES (decrements to 0 as the transfer progresses) can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

9.3.5.3 Preemption Status

Preemption is only available when fixed arbitration is selected for group and channel arbitration modes. A preemptable situation is when a preempt-enabled channel is running and a higher priority request becomes active. When the DMA_ENGINE is not operating in fixed group, fixed channel arbitration mode, the determination of the relative priority of the actively running and the outstanding requests become undefined. Channel and/or group priorities are treated as equal (or more exactly, constantly rotating) when round-robin arbitration mode is selected.

The TCD.ACTIVE bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one iteration of the major loop. Two TCD.ACTIVE bits set at the same time in the overall TCD map indicates a higher priority channel is actively preempting a lower priority channel.

The worst case latency when switching to a preempt channel is the summation of:

- Arbitration latency (2 cycles)
- Bandwidth control stalls (if enabled)
- The time to execute two read/write sequences (including AHB bus holds; a system dependency driven by the slave devices or the crossbar)

9.3.6 Channel Linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD.START bit of another channel (or itself) that initiates a service request for that channel. This operation is automatically performed by the DMA_ENGINE at the conclusion of the major or minor loop when properly enabled.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD.CITER.E_LINK field is used to determine whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made.

For example, with the initial fields of:

```
TCD.citer.e_link= 1
TCD.citer.linkch= 0xC
TCD.citer value= 0x4
TCD.major.e_link= 1
TCD.major.linkch= 0x7
```

Execute as:

1. Minor loop done → set channel 12 TCD.START bit.
2. Minor loop done → set channel 12 TCD.START bit.
3. Minor loop done → set channel 12 TCD.START bit.
4. Minor loop done, major loop done → set channel 7 TCD.START bit.

When minor loop linking is enabled (TCD.CITER.E_LINK = 1), the TCD.CITER field uses a 9-bit vector to form the current iteration count.

When minor loop linking is disabled (TCD.CITER.E_LINK = 0), the TCD.CITER field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD.CITER.LINKCH field are concatenated onto the CITER value to increase the range of the CITER.

NOTE

The TCD.CITER.E_LINK bit and the TCD.BITER.E_LINK bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

9.3.7 Dynamic Programming

This section provides recommended methods to change the programming model during channel execution.

9.3.7.1 Dynamic Priority Changing

The following two options are recommended for dynamically changing channel priority levels:

- Switch to round-robin channel arbitration mode, change the channel priorities, and then switch back to fixed arbitration mode
- Disable all the channels within a group, change the channel priorities within that group only, and then enable the appropriate channels.

The following two options are available for dynamically changing group priority levels:

- Switch to round-robin group arbitration mode, change the group priorities, and then switch back to fixed arbitration mode,
- Disable all channels, change the group priorities, and then enable the appropriate channels.

9.3.7.2 Dynamic Channel Linking and Dynamic Scatter/Gather

Dynamic channel linking and dynamic scatter/gather is the process of changing the TCD.MAJOR.E_LINK or TCD.E_SG bits during channel execution. These bits are read from the TCD local memory at the end of channel execution, allowing you to enable either feature during channel execution.

Because you can change the configuration during execution, a coherency model is needed. Consider the scenario where you attempt to execute a dynamic channel link by enabling the TCD.MAJOR.E_LINK bit at the same time the DMA_ENGINE is retiring the channel. The TCD.MAJOR.E_LINK would be set in

the programmer’s model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link or dynamic scatter/gather request:

1. Set the TCD.MAJOR.E_LINK bit.
2. Read back the TCD.MAJOR.E_LINK bit.
3. Test the TCD.MAJOR.E_LINK request status.
4. If the bit is set, the dynamic link attempt was successful.
5. If the bit is cleared, the attempted dynamic link did not succeed. The channel was already retiring.

This same coherency model is true for dynamic scatter/gather operations. For both dynamic requests, the TCD local memory controller forces the TCD.MAJOR.E_LINK and TCD.E_SG bits to 0 on any writes to a channel’s TCD.WORD7 after that channel’s TCD.DONE bit is set, indicating the major loop is complete.

NOTE

Clear the TCD.DONE bit before writing the TCD.MAJOR.E_LINK or TCD.E_SG bits. The TCD.DONE bit is cleared automatically by the DMA_ENGINE after a channel begins execution.

Chapter 10
Display Interface Unit (DIU)

10.1 Introduction

The Display Interface Unit (DIU) is a display controller designed to manage a thin film transistor liquid crystal display (TFT-LCD). Besides generating all the signals required to drive the display, the DIU manages real-time blending of as many as three planes onto the display.

10.1.1 Features

- Display color depth as high as 24 bits per pixel (bpp)
- Display interfaces: parallel TTL
- Maximum of three physical input planes
  - Memory writeback mode to store intermediate results, extending the number of graphics planes
- RGB and 256-level grayscale input pixel formats
- Programmable bit order definition as high as 8 bits per component
- Hardware cursor: 32 × 32 pixels, 16 bpp
- α-blending range as high as 256 levels
- Chroma Keying: Selectable by range
- Independent programmable gamma adjustments for each color component

10.1.2 Modes of Operation

The DIU has five modes of operation:

- Mode 0: DIU OFF. In this mode, the DIU is disabled.
- Mode 1: All three planes output to display. This is the typical operating mode of the DIU.
- Mode 2: Plane 1 to display, Planes 2 and 3 written back to memory. This mode is used to display a plane while processing (and writing back to memory) the data for other planes.
- Mode 3: All three planes written back to memory. This mode is used to process and write back to memory the data for the planes without displaying any of them.
- Mode 4: Color Bar Generation. This is a debug mode to check the operation of the DIU without the need for setting up the display memory structures in memory.

These modes are set by programming the DIU_MODE register. See Section 10.3.3.8, “DIU Mode of Operation Register (DIU_MODE),” for more information.

10.2 External Signal Description

Table 10-1 describes the DIU input and output signals, the meaning of their different states, and relative timing information for assertion and negation.

Table 10-1. Display Interface Detailed Signal Descriptions

Signal	I/O	Description	
DIU_CLK[1]	O	Pixel clock. This signals is used to drive the display panel.	
DIU_VSYNC	O	Vertical synchronizing signal. This signal indicates the beginning of a new frame. This signal may alternately be programmed to output a composite sync (CSYNC) signal by programming SYN_POL[BP_VS]. See Section 10.3.3.16, "Synchronization Signals Polarity Register (SYN_POL)," for more information. The composite sync signal combines the horizontal and vertical synchronizing signals to form a composite synchronizing signal. It includes both the HSYNC pulse and the VSYNC pulse. The default output is DIU_VSYNC.	
		State Meaning	Asserted at the beginning of a new frame.
		Timing	Asserted with the first cycle of the frame period. The length of the pulse is programmable.
DIU_HSYNC	O	Horizontal synchronizing signal. This signal indicates the beginning of a new line. This signal may alternately be programmed to output a composite sync (CSYNC) signal by programming SYN_POL[BP_HS]. See Section 10.3.3.16, "Synchronization Signals Polarity Register (SYN_POL)," for more information. The composite sync signal combines the horizontal and vertical synchronizing signals to form a composite synchronizing signal. It includes both the HSYNC pulse and the VSYNC pulse. The default output is DIU_HSYNC.	
		State Meaning	Asserted at the beginning of a new line.
		Timing	Asserted with the first cycle of a new line. The length of the pulse is programmable.
DIU_DE	O	Data enable. This signal qualifies the data on the data output signals (DIU_LD)	
		State Meaning	Deasserted: DIU_LD data is not valid. Asserted: DIU_LD data is valid.
DIU_LD[23:0]	O	Data output signals. • DIU_LD[23:16] = Red[7:0]. DIU_LD[23] is the most significant bit, and DIU_LD[16] is the least significant bit of the Red component. • DIU_LD[15:8] = Green[7:0]. DIU_LD[15] is the most significant bit, and DIU_LD[8] is the least significant bit of the Green component. • DIU_LD[7:0] = Blue[7:0]. DIU_LD[7] is the most significant bit, and DIU_LD[0] is the least significant bit of the Blue component.	

1 Refer to the system clock chapter for details on this clock.

10.3 Memory Map and Register Definition

10.3.1 Memory Map

Table 10-2 shows the register memory map for the DIU memory controller.

Table 10-2. DIU memory map

Offset from DIU_BASE (0xFF40_2100)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x00	DESC_1—Plane 1 Area Descriptor Pointer Register	R/W	0x0000_0000	10.3.3.1/10-246
0x04	DESC_2—Plane 2 Area Descriptor Pointer Register	R/W	0x0000_0000	10.3.3.2/10-247
0x08	DESC_3—Plane 3 Area Descriptor Pointer Register	R/W	0x0000_0000	10.3.3.3/10-247
0x0C	GAMMA—Gamma Table Pointer Register	R/W	0x0000_0000	10.3.3.4/10-248
0x10	PALETTE—Palette Table Pointer Register	R/W	0x0000_0000	10.3.3.5/10-248
0x14	CURSOR—Cursor Bitmap Table Pointer Register	R/W	0x0000_0000	10.3.3.6/10-249
0x18	CURS_POS—Cursor Position Register	R/W	0x0000_0000	10.3.3.7/10-250
0x1C	DIU_MODE—DIU Mode of Operation Register	R/W	0x0000_0000	10.3.3.8/10-250
0x20	BGND—Background Color Register	R/W	0x0000_0000	10.3.3.9/10-251
0x24	BGND_WB—Background Color in Writeback Mode Register	R/W	0x0000_0000	10.3.3.10/10-251
0x28	DISP_SIZE—Display Size Register	R/W	0x0000_0000	10.3.3.11/10-252
0x2C	WB_SIZE—Writeback Plane Size Register	R/W	0x0000_0000	10.3.3.12/10-253
0x30	WB_MEM_ADDR—Writeback Plane Address Register	R/W	0x0000_0000	10.3.3.13/10-253
0x34	HSYN_PARA—Horizontal Sync Pulse Parameters Register	R/W	0x0000_0000	10.3.3.14/10-254
0x38	VSYN_PARA—Vertical Sync Pulse Parameters Register	R/W	0x0000_0000	10.3.3.15/10-254
0x3C	SYN_POL— Synchronization Signals Polarity Register	R/W	0x0000_0000	10.3.3.16/10-255
0x40	THRESHOLDS—Thresholds Register	R/W	0x8000_F800	10.3.3.17/10-256
0x44	INT_STATUS—Interrupt Status Register	R	0x0000_0000	10.3.3.18/10-256
0x48	INT_MASK—Interrupt Mask Register	R/W	0x0000_003F	10.3.3.19/10-257
0x4C	COLBAR_1—Color #1 in the Color Bar (Black) Register	R/W	0xFF00_0000	10.3.3.20.1/10-259
0x50	COLBAR_2—Color #2 in the Color Bar (Blue) Register	R/W	0xFF00_00FF	10.3.3.20.2/10-259
0x54	COLBAR_3—Color #3 in the Color Bar (Cyan) Register	R/W	0xFF00_FFFF	10.3.3.20.3/10-259
0x58	COLBAR_4—Color #4 in the Color Bar (Green) Register	R/W	0xFF00_FF00	10.3.3.20.4/10-260
0x5C	COLBAR_5—Color #5 in the Color Bar (Yellow) Register	R/W	0xFFFF_FF00	10.3.3.20.5/10-260
0x60	COLBAR_6—Color #6 in the Color Bar (Red) Register	R/W	0xFFFF_0000	10.3.3.20.6/10-260
0x64	COLBAR_7—Color #7in the Color Bar (Purple) Register	R/W	0xFFFF_00FF	10.3.3.20.7/10-261

Table 10-2. DIU memory map (Continued)

Offset from DIU_BASE (0xFF40_2100)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x68	COLBAR_8—Color #8 in the Color Bar (White) Register	R/W	0xFFFF_FFFF	10.3.3.20.8/10-261
0x6C	FILLING—Filling Status Register	R	0x0000_0000	10.3.3.21/10-261
0x70	PLUT—Priority Look Up Table Register	R/W	0x0000_0000	10.3.3.22/10-262
0x74–0xFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, "System Configuration and Memory Map (XLBMEN + Mem Map)."

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

3 In this column, the symbol "U" indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

10.3.2 Register Summary

Table 10-3. DIU Block Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DESC_1 0x00	R	DESC_1[31:16]															
	W																
	R	DESC_1[15: 0]															
	W																
DESC_2 0x04	R	DESC_2[31:16]															
	W																
	R	DESC_2[15: 0]															
	W																
DESC_3 0x08	R	DESC_3[31:16]															
	W																
	R	DESC_3[15: 0]															
	W																
GAMMA 0x0C	R	GAMMA[31:16]															
	W																
	R	GAMMA[15: 0]															
	W																

Table 10-3. DIU Block Register Summary (Continued)

Name		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
PALETTE 0x10	R	PALETTE[31:16]															
	W																
	R	PALETTE[15: 0]															
	W																
CURSOR 0x14	R	CURSOR[31:16]															
	W																
	R	CURSOR[15: 0]															
	W																
CURS_POS 0x18	R	0	0	0	0	0	0	CURSOR_Y[10: 0]									
	W																
	R	0	0	0	0	0	0	CURSOR_X[10: 0]									
	W																
DIU_MODE 0x1C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIU_MODE[2:0]		
	W																
BGND 0x20	R	0	0	0	0	0	0	0	0	BGND_R[7:0]							
	W																
	R	BGND_G[7:0]								BGND_B[7:0]							
	W																
BGND_WB 0x24	R	0	0	0	0	0	0	0	0	BGND_WB_R[7:0]							
	W																
	R	BGND_WB_G[7:0]								BGND_WB_B[7:0]							
	W																
DISP_SIZE 0x28	R	0	0	0	0	0	DELTA_Y[10: 0]										
	W																
	R	0	0	0	0	0	DELTA_X[10: 0]										
	W																
WB_SIZE 0x2C	R	0	0	0	0	0	DELTA_Y_WB[10: 0]										
	W																
	R	0	0	0	0	0	DELTA_X_WB[10: 0]										
	W																

Table 10-3. DIU Block Register Summary (Continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WB_MEM_ADDR 0x30	R	WB_MEM_ADDR[31:16]															
	W																
	R	WB_MEM_ADDR[15: 0]															
	W																
HSYN_PARA 0x34	R	BP_H[9:0]										0	PW_H[9:5]				
	W																
	R	PW_H[4:0]					0	FP_H[9:0]									
	W																
VSYN_PARA 0x38	R	BP_V[9:0]										0	PW_V[9:5]				
	W																
	R	PW_V[4:0]					0	FP_V[9:0]									
	W																
SYN_POL 0x3C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	BP_VS	BP_HS	INV_CS	INV_VS	INV_HS
	W																
THRESHOLDS 0x40	R	1	0	0	0	0	LS_BF_VS[10:0]										
	W																
	R	1	1	1	1	1	0	0	0	OUT_BUF_LOW[7:0]							
	W																
INT_STATUS 0x44	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	WB_PEND	LS_BF_VS	PARERR	UNDRUN	VSYNC_WB	VSYNC
	W																
INT_MASK 0x48	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	M_WB_PEND	M_LS_BF_VS	M_PARERR	M_UNDRUN	M_VSYN_WB	M_VSYNC
	W																

Table 10-3. DIU Block Register Summary (Continued)

Name		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
COLBAR_1 0x4C	R	1	1	1	1	1	1	1	1	COLBAR_1_R[7:0]							
	W																
	R	COLBAR_1_G[7:0]								COLBAR_1_B[7:0]							
	W																
COLBAR_2 0x50	R	1	1	1	1	1	1	1	1	COLBAR_2_R[7:0]							
	W																
	R	COLBAR_2_G[7:0]								COLBAR_2_B[7:0]							
	W																
COLBAR_3 0x54	R	1	1	1	1	1	1	1	1	COLBAR_3_R[7:0]							
	W																
	R	COLBAR_3_G[7:0]								COLBAR_3_B[7:0]							
	W																
COLBAR_4 0x58	R	1	1	1	1	1	1	1	1	COLBAR_4_R[7:0]							
	W																
	R	COLBAR_4_G[7:0]								COLBAR_4_B[7:0]							
	W																
COLBAR_5 0x5C	R	1	1	1	1	1	1	1	1	COLBAR_5_R[7:0]							
	W																
	R	COLBAR_5_G[7:0]								COLBAR_5_B[7:0]							
	W																
COLBAR_6 0x60	R	1	1	1	1	1	1	1	1	COLBAR_6_R[7:0]							
	W																
	R	COLBAR_6_G[7:0]								COLBAR_6_B[7:0]							
	W																
COLBAR_7 0x64	R	1	1	1	1	1	1	1	1	COLBAR_7_R[7:0]							
	W																
	R	COLBAR_7_G[7:0]								COLBAR_7_B[7:0]							
	W																
COLBAR_8 0x68	R	1	1	1	1	1	1	1	1	COLBAR_8_R[7:0]							
	W																
	R	COLBAR_8_G[7:0]								COLBAR_8_B[7:0]							
	W																

Table 10-3. DIU Block Register Summary (Continued)

Name		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
FILLING 0x6C	R	0	0	0	0	0	0	FILLING_OBF[9:0]									
	W																
	R	FILLING_WB[3:0]				FILLING_P3[3:0]				FILLING_P2[3:0]				FILLING_P1[3:0]			
	W																
PLUT 0x70	R	PRIORITY_7[3:0]				PRIORITY_6[3:0]				PRIORITY_5[3:0]				PRIORITY_4[3:0]			
	W																
	R	PRIORITY_3[3:0]				PRIORITY_2[3:0]				PRIORITY_1[3:0]				PRIORITY_0[3:0]			
	W																

10.3.3 Register Descriptions

10.3.3.1 Plane 1 Area Descriptor Pointer Register (DESC_1)

The Plane 1 Area Descriptor Pointer (DESC_1) register is the plane 1 Area Descriptor (AD) pointer. It sets the base address of the first plane 1 AD. This address must be 64-bit boundary aligned (set the lowest 3 bits to 0).

Address: Base + 0x00 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DESC_1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DESC_1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-1. Plane 1 Area Descriptor Pointer Register (DESC_1)

Table 10-4. DESC_1 field descriptions

Field	Description
DESC_1	Plane 1 area descriptor pointer DESC_1 = 0x0000_0000 means that no AD is available for this plane.

10.3.3.2 Plane 2 Area Descriptor Pointer Register (DESC_2)

The Plane 2 Area Descriptor Pointer (DESC_2) register is the plane 2 Area Descriptor (AD) pointer. It sets the base address of the first plane 2 AD. This address must be 64-bit boundary aligned (set the lowest 3 bits to 0).

Address: Base + 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DESC_2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DESC_2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-2. Plane 2 Area Descriptor Pointer Register (DESC_2)

Table 10-5. DESC_2 field descriptions

Field	Description
DESC_2	Plane 2 area descriptor pointer. DESC_2 = 0x0000_0000 means that no AD is available for this plane.

10.3.3.3 Plane 3 Area Descriptor Pointer Register (DESC_3)

The Plane 3 Area Descriptor Pointer (DESC_3) register is the plane 3 Area Descriptor (AD) pointer. It sets the base address of the first plane 3 AD. This address must be 64-bit boundary aligned (set the lowest 3 bits to 0).

Address: Base + 0x08 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DESC_3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DESC_3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-3. Plane 3 Area Descriptor Pointer Register (DESC_3)

Table 10-6. DESC_3 field descriptions

Field	Description
DESC_3	Plane 3 area descriptor pointer. DESC_3 = 0x0000_0000 means that no AD is available for this plane.

10.3.3.4 Gamma Table Pointer Register (GAMMA)

Gamma Table Pointer Register (GAMMA) sets the base address to the GAMMA table in memory. Writing to this register causes the DIU to load the new GAMMA table from there. This address must be 64-bit boundary aligned (set the lowest 3 bits to 0). A 32-byte boundary-aligned address is more efficient.

Address: Base + 0x0C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	GAMMA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GAMMA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-4. GAMMA Register

Table 10-7. GAMMA field descriptions

Field	Description
GAMMA	Base address to the GAMMA table in memory.

10.3.3.5 Palette Table Pointer Register (PALETTE)

The Palette Table Pointer Register (PALETTE) sets the base address to the Palette table in memory. Writing to this register causes the DIU to load the new Palette table from there. This address must be 64-bit boundary aligned (set the lowest 3 bits to 0). A 32-byte boundary aligned address is more efficient.

Address: Base + 0x10 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PALETTE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PALETTE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-5. Palette Table Pointer Register (PALETTE)

Table 10-8. PALETTE field descriptions

Field	Description
PALETTE	Base address to the PALETTE table in memory.

10.3.3.6 Cursor Bitmap Table Pointer Register (CURSOR)

The Cursor Bitmap Table Pointer Register (CURSOR) sets the base address to the CURSOR bitmap in memory. Writing to this register causes the DIU to load the new CURSOR bitmap from there. This address must be 64-bit boundary aligned (set the lowest 3 bits to 0). A 32-byte boundary aligned address is more efficient.

Address: Base + 0x14 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CURSOR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CURSOR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-6. Cursor Bitmap Table Pointer Register (CURSOR)

Table 10-9. CURSOR field descriptions

Field	Description
CURSOR	Base address to the CURSOR bitmap in memory.

10.3.3.7 Cursor Position Register (CUR_POS)

The Cursor Position Register (CUR_POS) sets the position of the cursor in the display. Table 10-10 shows its field descriptions.

Address: Base + 0x18 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	CURSOR_Y										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CURSOR_X										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-7. Cursor Position Register (CUR_POS)

Table 10-10. CUR_POS field descriptions

Field	Description
CURSOR_Y	Vertical position of the cursor (in pixels), from the top-left corner.
CURSOR_X	Horizontal position of the cursor (in pixels), from the top-left corner.

10.3.3.8 DIU Mode of Operation Register (DIU_MODE)

The DIU Mode of Operation Register (DIU_MODE) sets the operation mode of the DIU. See Table 10-11 for its field descriptions.

Address: Base + 0x1C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIU_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-8. DIU Mode of Operation Register (DIU_MODE)

Table 10-11. DIU_MODE field descriptions

Field	Description
DIU_MODE	DIU Operation Mode 000 Encoding Mode 0: DIU OFF. 001 Encoding Mode 1: All three planes output to display. 010 Encoding Mode 2: Plane 1 to display, planes 2 and 3 written back to memory. 011 Encoding Mode 3: All three planes written back to memory. 100 Encoding Mode 4: Color Bar Generation. All other encodings are reserved **Note:** Writing a reserved value is blocked and does not affect the register value.

10.3.3.9 Background Register (BGND)

The Background (BGND) register sets the default background color for plane 1 (for mode 1 or 2). This is the color used to fill the areas for which no data is assigned in the area descriptor.

Address: Base + 0x20 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	BGND_R							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BGND_G								BGND_B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-9. Background Register (BGND)

Table 10-12. BGBD field descriptions

Field	Description
BGND_R	BGND_R represents the red component of the background.
BGND_G	BGND_G represents the green component of the background.
BGND_B	BGND_B represents the blue component of the background.

10.3.3.10 Background Writeback Register (BGND_WB)

The Background Writeback (BGND_WB) register sets the default background color for the writeback planes (plane 2 in mode 2 or plane 1 in mode 3). This is the color used to fill the areas for which no data is assigned in the area descriptor. The bottom plane is expanded to the writeback frame size with the background color, and then blended with the top plane(s).

Address: Base + 0x24 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	BGND_WB_R							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BGND_WB_G								BGND_WB_B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-10. Background Writeback Register (BGND_WB)

Table 10-13. BGND_WB field descriptions

Field	Description
BGND_WB_R	BGND_WB_R represents the red component of the background for the writeback planes.
BGND_WB_G	BGND_WB_G represents the green component of the background for the writeback planes.
BGND_WB_B	BGND_WB_B represents the blue component of the background for the writeback planes.

10.3.3.11 Display Size Register (DISP_SIZE)

The Display Size Register (DISP_SIZE) sets the display size (in pixels).

Address: Base + 0x28 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	DELTA_Y										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	DELTA_X										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-11. Display Size Register (DISP_SIZE)

Table 10-14. DISP_SIZE field descriptions

Field	Description
DELTA_Y	DELTA_Y represents the vertical resolution.
DELTA_X	DELTA_X represents the horizontal resolution.

10.3.3.12 Writeback Plane Size Register (WB_SIZE)

The Writeback Plane Size Register (WB_SIZE) register sets the writeback frame size (in pixels).

Address: Base + 0x2C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	DELTA_Y_WB										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	DELTA_X_WB										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-12. Writeback Plane Size Register (WB_SIZE)

Table 10-15. WB_SIZE field descriptions

Field	Description
DELTA_Y_WB	DELTA_Y_WB represents the vertical resolution.
DELTA_X_WB	DELTA_X_WB represents the horizontal resolution.

10.3.3.13 Writeback Plane Address Register (WB_MEM_ADDR)

The Writeback Plane Address Register (WB_MEM_ADDR) sets the base address where the writeback frame is written to in the memory. A write to this register triggers a writeback frame refresh. This address must be 64-bit boundary aligned (set the lowest 3 bits to 0). A 32-byte boundary aligned address is more efficient.

Address: Base + 0x30 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WB_MEM_ADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WB_MEM_ADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-13. Writeback Plane Address Register (WB_MEM_ADDR)

Table 10-16. WB_MEM_ADDR field descriptions

Field	Description
WB_MEM_ ADDR	Base address where the writeback frame is written to in the memory.

10.3.3.14 Horizontal Sync Pulse Parameters Register (HSYN_PARA)

The Horizontal Sync Pulse Parameters Register (HSYN_PARA) sets timing parameters related to the horizontal synchronization signal generation. See Figure 10-49, the display timing diagrams, for detailed signal meaning.

Address: Base + 0x34 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BP_H										0	PW_H				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PW_H					0	FP_H									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-14. Horizontal Sync Pulse Parameters Register (HSYN_PARA)

Table 10-17. HSYN_PARA field descriptions

Field	Description
BP_H	HSYNC back-porch pulse width (in pixel clock cycles). It can be 0.
PW_H	HSYNC active pulse width (in pixel clock cycles). It must be greater than or equal to 1.
FP_H	HSYNC front-porch pulse width (in pixel clock cycles). It can be 0.

10.3.3.15 Vertical Sync Pulse Parameters Register (VSYN_PARA)

The Vertical Sync Pulse Parameters Register (VSYN_PARA) sets timing parameters related to the vertical synchronization signal generation. See Figure 10-50, the display timing diagram, for detailed signal meaning.

Address: Base + 0x38 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BP_V										0	PW_V				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PW_V					0	FP_V									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-15. Vertical Sync Pulse Parameters Register (VSYN_PARA)

Table 10-18. VSYN_PARA field descriptions

Field	Description
BP_V	VSYNC back-porch pulse width (in HSYNC signal cycles) It can be 0.
PW_V	VSYNC active pulse width (in HSYNC signal cycles). **Note:** This bit must be set to a non-zero value for the display to operate.
FP_V	VSYNC front-porch pulse width (in HSYNC signal cycles). It can be 0.

10.3.3.16 Synchronization Signals Polarity Register (SYN_POL)

The Synchronization Signals Polarity Register (SYN_POL) selects polarity for the horizontal sync (HSYNC), vertical sync (VSYNC), and composite sync (CSYNC) synchronization signals, and controls bypassing the HSYNC or VSYNC signals with the CSYNC signal.

Address: Base + 0x3C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	BP_VS	BP_HS	INV_CS	INV_VS	INV_HS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-16. Synchronization Signals Polarity Register (SYN_POL)

Table 10-19. SYN_POL field descriptions

Field	Description
BP_VS	Bypass Vertical Synchronize Signal (internal pin muxing) 0 Do not bypass the VSYNC signal output. 1 CSYNC bypasses the VSYNC signal and outputs CSYNC instead of VSYNC.
BP_HS	Bypass Horizontal Synchronize Signal (internal pin muxing) 0 Do not bypass the HSYNC signal output. 1 CSYNC bypasses the HSYNC signal, and outputs CSYNC instead of HSYNC.
INV_CS	Invert Composite Synchronize Signal 0 CSYNC signal is not inverted and is active HIGH. 1 CSYNC signal is inverted and is active LOW.
INV_VS	Invert Vertical Synchronize Signal 0 VSYNC signal is not inverted and is active HIGH. 1 VSYNC signal is inverted and is active LOW.
INV_HS	Invert Horizontal Synchronize Signal 0 HSYNC signal is not inverted and is active HIGH. 1 HSYNC signal is inverted and is active LOW.

10.3.3.17 Thresholds Register (THRESHOLDS)

The Thresholds Register (THRESHOLDS) sets threshold values related to DIU operations.

Address: Base + 0x40 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	0	0	0	0	LS_BF_VS										
W																
Reset	1[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	1	1	1	1	0	0	0	OUT_BUF_LOW							
W																
Reset	1[1]	1[1]	1[1]	1[1]	1[1]	0	0	0	0	0	0	0	0	0	0	0

Figure 10-17. Thresholds Register (THRESHOLDS)

[1] These reserved bits are set to 1 on reset and must always be set to 1.

Table 10-20. THRESHOLDS field descriptions

Field	Description
LS_BF_VS	Lines Before VSYNC Threshold. This threshold value generates the LS_BF_VS interrupt status. It sets the number of lines ahead of the vertical front porch (FP_V) when the interrupt is generated.
OUT_BUF_LOW	Output Buffer Filling Low Threshold (in pixels). Generates the buffer underrun exception. An underrun exception is generated if the display needs data, and the output buffer filling is lower than or equal to the OUT_BUF_LOW threshold.

10.3.3.18 Interrupt Status Register (INT_STATUS)

The Interrupt Status Register (INT_STATUS) indicates the interrupt status. DIU has only one interrupt signal. The CPU reads the INT_STATUS register to decide which exception occurred when an interrupt is detected. The read operation also clears the register.

Address: Base + 0x44 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	WB_PEND	LS_BF_VS	PAR ERR	UND RUN	VSYNC_WB	V SYNC
W											r1c	r1c	r1c	r1c	r1c	r1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-18. Interrupt Status Register (INT_STATUS)

Table 10-21. INT_STATUS field descriptions

Field	Description
WB_PEND	Writeback Pending Interrupt. If enabled, this interrupt is generated in mode 2 (the combined display and writeback mode) when a writeback operation does not complete before the display frame parallel to it. This is considered an error. The user can select whether to redo the writeback frame, or ignore it. This interrupt is disabled by default, and is enabled by clearing INT_MASK[M_WB_PEND].
LS_BF_VS	Lines Before VSYNC interrupt. If enabled, this interrupt is generated at the number of lines specified by THRESHOLDS[LS_BF_VS] ahead of the vertical front porch (FP_V). This interrupt is disabled by default, and is enabled by clearing INT_MASK[M_LS_BF_VS].
PARERR	Display parameter error interrupt. If enabled, this interrupt is generated if the user sets incorrect display parameters. This interrupt is disabled by default, and is enabled by clearing INT_MASK[M_PARERR].
UNDRUN	Underrun exception interrupt. If enabled, this interrupt is generated when the display needs data, and the output buffer filling is lower than or equal to the OUT_BUF_LOW threshold. This interrupt is disabled by default, and is enabled by clearing INT_MASK[M_UNDRUN].
VSYNC_WB	Vertical Synchronize Interrupt for writeback operation. If enabled, this interrupt is generated at the end of a writeback frame. Used in mode 2 and 3 only. This interrupt is disabled by default, and is enabled by clearing INT_MASK[M_VSYN_WB].
VSYNC	Vertical synchronization interrupt. If enabled, this interrupt is generated at the beginning of a frame. This interrupt is disabled by default, and is enabled by clearing INT_MASK[M_VSYNC].

10.3.3.19 Interrupt Mask Register (INT_MASK)

The Interrupt Mask Register (INT_MASK) enables or masks interrupts corresponding to the interrupt status bits in the INT_STATUS register.

Address: Base + 0x48 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	M_WB_PEND	M_LS_BF_VS	M_PARERR	M_UNDRUN	M_VSYC_WB	M_VSYNC
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 10-19. Interrupt Mask Register (INT_MASK)

Table 10-22. INT_MASK field descriptions

Field	Description
M_WB_PEND	Writeback Pending Interrupt Mask Bit. 0 Interrupt is enabled. 1 Interrupt is masked (the default setting).
M_LS_BF_VS	Lines Before VSYNC Interrupt Mask Bit. 0 Interrupt is enabled. 1 Interrupt is masked (the default setting).
M_PARERR	Display Parameter Error Interrupt Mask Bit. 0 Interrupt is enabled. 1 Interrupt is masked (the default setting).
M_UNDRUN	Underrun Exception Interrupt Mask Bit. 0 Interrupt is enabled. 1 Interrupt is masked (the default setting).
M_VSYN_WB	Vertical Synchronize Interrupt for Writeback Mask Bit. 0 Interrupt is enabled. 1 Interrupt is masked (the default setting).
M_VSYNC	Vertical Synchronization Interrupt Mask Bit. 0 Interrupt is enabled. 1 Interrupt is masked (the default setting).

10.3.3.20 COLBAR Registers

The COLBAR registers are used to generate color bars in functional test mode. Eight different pixel values are taken as input data to display eight color bars on the display. After reset, they take the default values shown in Table 10-23.

Table 10-23. COLBAR_*n* field descriptions

COLBAR_*n*	Default Value	Color	COLBAR_*n*	Default Value	Color
COLBAR_1	0xFF00_0000	Black	COLBAR_5	0xFFFF_FF00	Yellow
COLBAR_2	0xFF00_00FF	Blue	COLBAR_6	0xFFFF_0000	Red
COLBAR_3	0xFF00_FFFF	Cyan	COLBAR_7	0xFFFF_00FF	Purple
COLBAR_4	0xFF00_FF00	Green	COLBAR_8	0xFFFF_FFFF	White

Programming the COLBAR registers at the middle of a frame affects the display immediately, so the user should reprogram them after VSYNC interrupt is detected.

Table 10-24. COLBAR_*n* field descriptions

Field	Description
COLBAR_*n*_R	Red component of RGB color.
COLBAR_*n*_G	Green component of RGB color.
COLBAR_*n*_B	Blue component of RGB color.

10.3.3.20.1 Color #1 in the Color Bar (Black) Register (COLBAR_1)

Address: Base + 0x4C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_1_R							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_1_G								COLBAR_1_B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-20. COLBAR_1 Register (Black)

10.3.3.20.2 Color #2 in the Color Bar (Blue) Register (COLBAR_2)

Address: Base + 0x50 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_2_R							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_2_G								COLBAR_2_B							
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Figure 10-21. COLBAR_2 Register (Blue)

10.3.3.20.3 Color #3 in the Color Bar (Cyan) Register (COLBAR_3)

Address: Base + 0x54 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_3_R							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_3_G								COLBAR_3_B							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 10-22. COLBAR_3 Register (Cyan)

10.3.3.20.4 Color #4 in the Color Bar (Green) Register (COLBAR_4)

Address: Base + 0x58 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_4_R							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_4_G								COLBAR_4_B							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Figure 10-23. COLBAR_4 Register (Green)

10.3.3.20.5 Color #5 in the Color Bar (Yellow) Register (COLBAR_5)

Address: Base + 0x5C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_5_R							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_5_G								COLBAR_5_B							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Figure 10-24. COLBAR_5 Register (Yellow)

10.3.3.20.6 Color #6 in the Color Bar (Red) Register (COLBAR_6)

Address: Base + 0x60 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_6_R							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_6_G								COLBAR_6_B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-25. COLBAR_6 Register (Red)

10.3.3.20.7 Color #7 in the Color Bar (Purple) Register (COLBAR_7)

Address: Base + 0x64 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_7_R							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_7_G								COLBAR_7_B							
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Figure 10-26. COLBAR_7 Register (Purple)

10.3.3.20.8 Color #8 in the Color Bar (White) Register (COLBAR_8)

Address: Base + 0x68 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	1	1	1	1	COLBAR_8_R							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COLBAR_8_G								COLBAR_8_B							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 10-27. COLBAR_8 Register (White)

10.3.3.21 Filling Status Register (FILLING)

The Filling Status Register (FILLING) is a read-only register that indicates current filling status of the input and output buffers. It is useful for debug purpose.

Address: Base + 0x6C Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	FILLING_OBF									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FILLING_WB				FILLING_P3				FILLING_P2				FILLING_P1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-28. Filling Status Register (FILLING)

Table 10-25. FILLING field descriptions

Field	Description
FILLING_OBF	Filling status of the output buffer (in pixels, 24 bits per pixel).
FILLING_WB	Filling status of the writeback pixel buffer (number of filled buffers out of the eight 256-byte buffers).
FILLING_P3	Filling status of plane 3 input pixel buffer (number of filled buffers out of the eight 256-byte buffers).
FILLING_P2	Filling status of plane 2 input pixel buffer (number of filled buffers out of the eight 256-byte buffers).
FILLING_P1	Filling status of plane 1 input pixel buffer (number of filled buffers out of the 8 256-byte buffers).

10.3.3.22 Priority Look Up Table Register (PLUT)

The Priority Look Up Table Register (PLUT), shown in Figure 10-29, determines the priority of the DIU transactions relative to other initiators on the DDR DRAM buses. The PLUT register includes the eight Priority Look Up Table components. A 4-bit priority output is selected from this look up table according to the input buffer filling status. This register must be configured so that the DIU can escalate its priority dynamically. See Section 10.4.12, “Dynamic Priority Generation,” for details on how to configure it.

Address: Base + 0x70 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRIORITY_7				PRIORITY_6				PRIORITY_5				PRIORITY_4			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRIORITY_3				PRIORITY_2				PRIORITY_1				PRIORITY_0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-29. Priority Look Up Table Register (PLUT)

Table 10-26. PLUT field descriptions

Field	Description
PRIORITY_7	Priority Look Up Table component 7. PRIORITY_7 is the lowest priority.
PRIORITY_6	Priority Look Up Table component 6.
PRIORITY_5	Priority Look Up Table component 5.
PRIORITY_4	Priority Look Up Table component 4.
PRIORITY_3	Priority Look Up Table component 3.
PRIORITY_2	Priority Look Up Table component 2.
PRIORITY_1	Priority Look Up Table component 1.
PRIORITY_0	Priority Look Up Table component 0. PRIORITY_0 is the highest priority.

10.4 Functional Description

The DIU does not have internal frame buffers. It reads the data from the main memory (DDR DRAM) at the same rate it refreshes the display.

Besides generating all the signals required to drive the display, the DIU manages real time blending of as many as three planes onto the display. Alpha blending is performed between the planes. Chroma key support is also present to help relieve the host processor from all the computational and bandwidth consuming blending tasks while simultaneously allowing the users to maintain the graphics quality required by many applications.

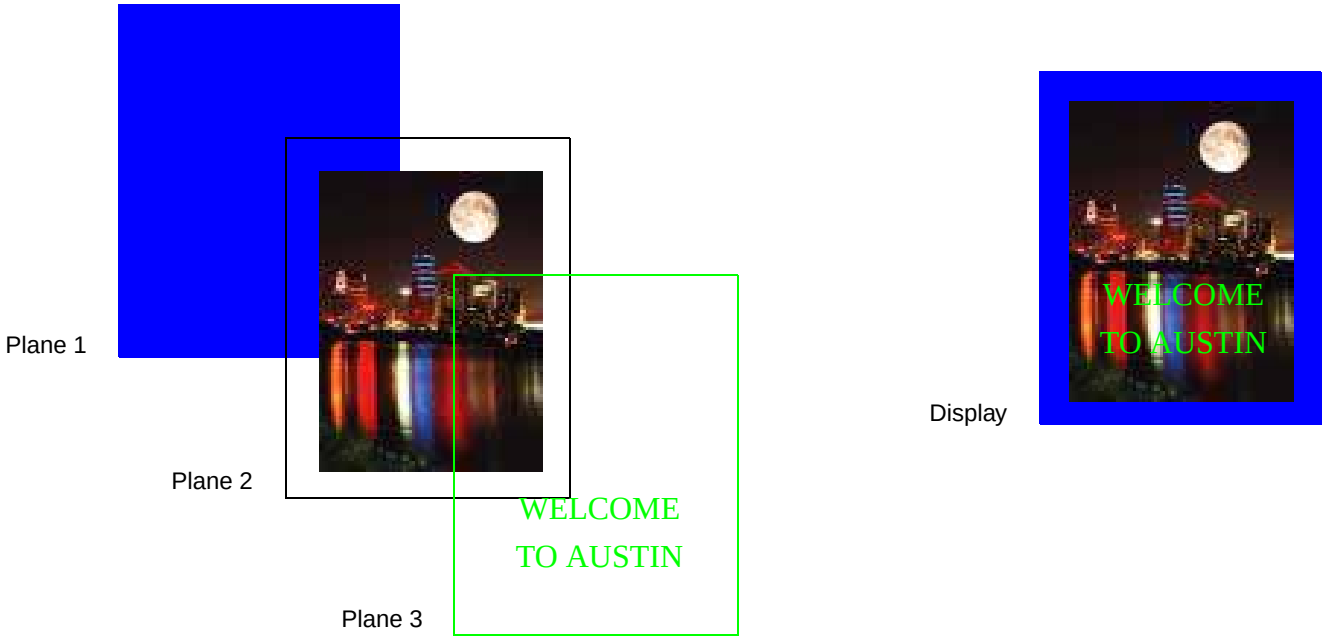


Figure 10-31. Three Plane Blending

10.4.1 Area Descriptor

The area descriptor (AD) defines each area to be displayed on a plane. A plane can display more than one area as long as the areas do not share a scan line.

The areas (for a plane) must be sorted in vertical order from top to bottom. The area descriptor is set up in the system's main memory (DDR DRAM) and then retrieved by the DIU directly from there.

Change the displayed data between frames by changing the data in the current area descriptor or create a new one and change the pointer in the DIU while keeping the previous one for future use or reference. It is always assumed that the bitmaps are stored pixel by pixel in memory, starting from the top-left most pixel in the image and continued sequentially until the last pixel (the bottom-right most pixel) by scanning the image always from left to right and top to bottom.

Figure 10-32 and Figure 10-33 show graphical representations of the area descriptor parameters that define an area. Figure 10-32 shows the parameters that specify how the source bitmap located in memory is interpreted by the DIU. You might want to display only a limited area of the bitmap by specifying an area of interest (AOI) as a subset of this bitmap.

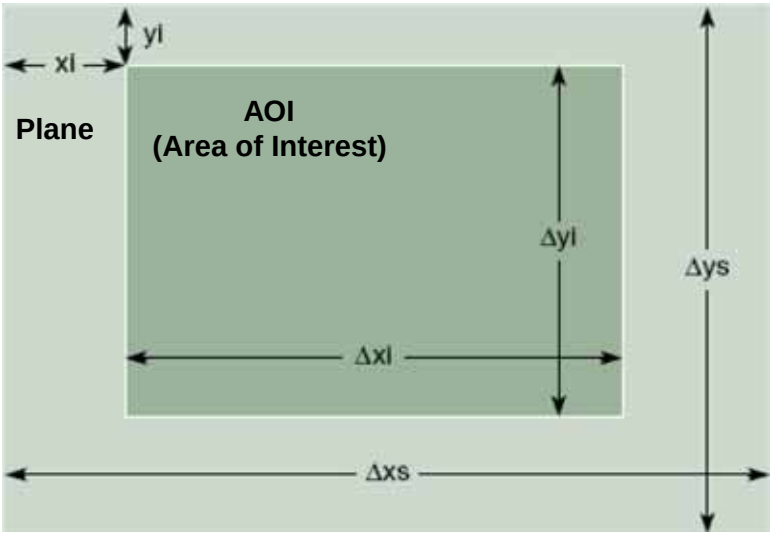


Figure 10-32. Source Bitmap Parameters

The complete source bitmap is specified by its starting address in memory, its pixel format, and its dimensions. The subset to be displayed is defined by its relative position to the beginning of the bitmap (top left corner) and its own size, which can be as large as the original image for the case where the whole image is to be displayed. The DIU automatically fetches the data, optimizing the accesses to minimize the bandwidth consumed by the operation.

There are some limitations on the size of an AOI, which include:

- The height and width of an AOI (Δyi, Δxi) must be greater than or equal to two pixels.
- The first AOI of a plane must be greater than or equal to 32 bytes of pixel data.

Figure 10-33 shows the parameters that specify how the image is to be displayed.

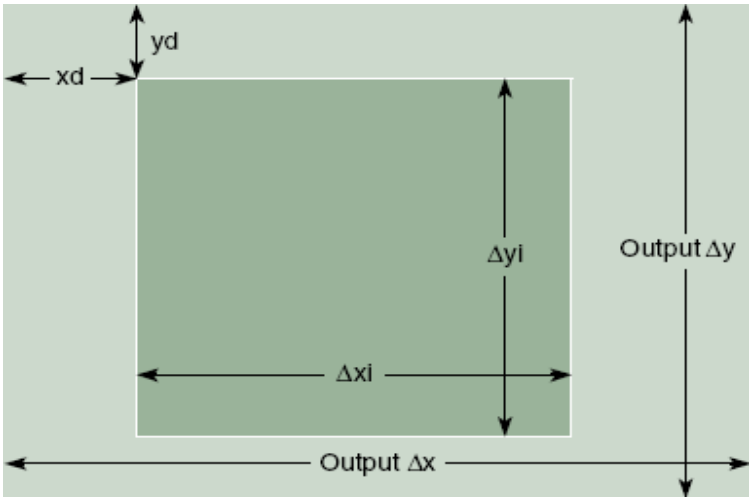


Figure 10-33. Display Parameters

Two parameters determine how this image is to be displayed. The first is its relative position with respect to the top-left corner of the display, and the second is its orientation (the source image can be flipped on its x-axis or y-axis, but not both simultaneously).

Since the size of the display is common to all area descriptors of all planes, it is not part of the area descriptor.

10.4.2 Area Descriptor Format

Each AD is composed of a ten-word data structure. The general format for an AD is shown in Table 10-28.

Table 10-28. Area Descriptor Format

Offset from the start address of the area descriptor table in memory	General Format
0x00	Word 0—Pixel format
0x04	Word 1—Bitmap address
0x08	Word 2—Source size/Global alpha
0x0C	Word 3—AOI size
0x10	Word 4—AOI offset
0x14	Word 5—Display offset
0x18	Word 6—Chroma key maximum
0x1C	Word 7—Chroma key minimum
0x20	Word 8—Next AD
0x24	Word 9—Reserved

The following sections describe the individual components that make up the area descriptor.

NOTE

The area descriptor uses little-endian byte ordering. Do a 32-bit word endian swap for each word before writing it to memory.

10.4.2.1 Area Descriptor Word 0–Pixel Format

Figure 10-34 shows the fields of AD Word 0, which defines the pixel format. Table 10-29 describes the pixel format fields.

Address: Base + 0xC8 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	BYTE_F	ALPHA_C			BLUE_C		GREEN_C		RED_C		PALETTE	PIXEL_S	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_3				COMP_2				COMP_1				COMP_0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-34. Area Descriptor Word 0 Pixel Format

Table 10-29. Area Descriptor Word 0–Pixel Format field descriptions

Field	Description
BYTE_F	Byte flip disable. When cleared, flips the byte order for the pixel data. See Section 10.4.3, “Pixel Structure,” for more information. 0 Bytes are flipped. 1 Bytes are not flipped.
ALPHA_C	Alpha component assignment. This field assigns the component number used for the alpha channel. 000 Component 0. 001 Component 1. 010 Component 2. 011 Component 3. 100 Global alpha. Default value is 011.
BLUE_C	Blue component assignment. This field assigns the component number used for the blue channel. 00 Component 0. 01 Component 1. 10 Component 2. 11 Component 3. Default value is 00.
GREEN_C	Green component assignment. This field assigns the component number used for the green channel. 00 Component 0. 01 Component 1. 10 Component 2. 11 Component 3. Default value is 01.

Table 10-29. Area Descriptor Word 0—Pixel Format field descriptions (Continued)

Field	Description
RED_C	Red Component assignment. This field assigns the component number used for the red channel. 00 Component 0. 01 Component 1. 10 Component 2. 11 Component 3. Default value is 10.
PALETTE	Palette mode. Determines whether palette mode is used. 0 Palette mode disabled. 1 Palette mode enabled.
PIXEL_S	Pixel size. Specifies the number of bytes per pixel. The actual number of bytes per pixel is PIXEL_S + 1. 00 1 byte per pixel. 01 2 bytes per pixel. 10 3 bytes per pixel. 11 4 bytes per pixel.
COMP_3	Number of bits for component 3. Valid range is 0 through 8.
COMP_2	Number of bits for component 2. Valid range is 0 through 8.
COMP_1	Number of bits for component 1. Valid range is 0 through 8.
COMP_0	Number of bits for component 0. Valid range is 0 through 8.

10.4.2.2 Area Descriptor Word 1–Bitmap Address

Figure 10-35 shows the fields of AD Word 1, which defines the bitmap address. Table 10-30 describes the bitmap address fields.

Address: Base + 0xCC Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BM_ADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BM_ADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-35. Area Descriptor Word 1—Bitmap Address

Table 10-30. Area Descriptor Word 1—Bitmap Address field descriptions

Field	Description
BM_ADDR	Bitmap address pointer. It points to the bitmap data in memory.

10.4.2.3 Area Descriptor Word 2–Source Size/Global Alpha

Figure 10-36 shows the fields of AD Word 2, which defines the source size and the global alpha value. Table 10-31 describes the source size/global alpha fields.

Address: Base + 0xD0 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	G_ALPHA								0	DELTA_YS						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DELTA_YS				0	DELTA_XS										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-36. Area Descriptor Word 2—Source Size/Global Alpha

Table 10-31. Area Descriptor Word 2—Source Size/Global Alpha field descriptions

Field	Description
G_ALPHA	Global alpha. Value used for the alpha channel for all pixels in the area when the data format does not include an alpha component or when the user wants to override the alpha values in the data (ALPHA_C = 100). Pixels to be displayed on Plane 1 ignore the alpha value.
DELTA_YS	The Δys parameter that defines the vertical size (in pixels) of the source bitmap.
DELTA_XS	The Δxs parameter that defines the horizontal size (in pixels) of the source bitmap.

10.4.2.4 Area Descriptor Word 3—AOI Size

Figure 10-37 shows the fields of AD Word 3, which defines the size of the area of interest. Table 10-32 describes the area of interest size fields.

Address: Base + 0xD4 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FLIP		0	0	0	DELTA_YI										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	DELTA_XI										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-37. Area Descriptor Word 3—AOI Size

Table 10-32. Area Descriptor Word 3—AOI Size field descriptions

Field	Description
FLIP	Area of interest image flip. Flips the bitmap image either horizontally or vertically within the area of interest. 00 Normal. 01 Flip image horizontally (about the y-axis). 10 Flip image vertically (about the x-axis). 11 Reserved.
DELTA_YI	The ΔYI parameter that defines the vertical size (in pixels) of the area of interest.

Table 10-32. Area Descriptor Word 3—AOI Size field descriptions (Continued)

Field	Description
DELTA_XI	The ΔXI parameter that defines the horizontal size (in pixels) of the area of interest.

10.4.2.5 Area Descriptor Word 4–AOI Offset

Figure 10-38 shows the fields of AD Word 4, which defines the offset for the area of interest. Table 10-33 describes the area of interest offset fields.

Address: Base + 0xD8 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	OFFSET_YI										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	OFFSET_XI										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-38. Area Descriptor Word 4—AOI Offset

Table 10-33. Area Descriptor Word 4—AOI Offset field descriptions

Field	Description
OFFSET_YI	The YI parameter that defines the vertical offset (in pixels) of the area of interest from the start of the source bitmap in memory.
OFFSET_XI	The XI parameter that defines the horizontal offset (in pixels) of the area of interest from the start of the source bitmap in memory.

10.4.2.6 Area Descriptor Word 5–Display Offset

Figure 10-39 shows the fields of AD Word 5, which defines the offset for the area of interest in the display. Table 10-34 describes the display offset fields.

Address: Base + 0xDC Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	OFFSET_YD										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W						OFFSET_XD										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-39. Area Descriptor Word 5—Display Offset

Table 10-34. Area Descriptor Word 5—Display Offset field descriptions

Field	Description
OFFSET_YD	The YD parameter that defines the vertical offset (in pixels) of the area of interest in the display.
OFFSET_XD	The XD parameter that defines the horizontal offset (in pixels) of the area of interest in the display.

10.4.2.7 Area Descriptor Word 6–Chroma Key Max

Figure 10-40 shows the fields of AD Word 6, which defines the maximum values for chroma key. See Section 10.4.6, “Chroma Keying,” for more information. Table 10-35 describes the chroma key max fields.

Address: Base + 0xE0 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	CKMAX_B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CKMAX_G								CKMAX_R							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-40. Area Descriptor Word 6—Chroma Key Max

Table 10-35. Area Descriptor Word 6—Chroma Key Max field descriptions

Field	Description
CKMAX_B	Chroma key maximum blue component value.
CKMAX_G	Chroma key maximum green component value.
CKMAX_R	Chroma key maximum red component value.

10.4.2.8 Area Descriptor Word 7–Chroma Key Minimum

Figure 10-41 shows the fields of AD Word 7, which defines the minimum values for chroma key. See Section 10.4.6, “Chroma Keying,” for more information. Table 10-36 describes the chroma key minimum fields.

Address: Base + 0xE4 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	CKMIN_B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CKMIN_G								CKMIN_R							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-41. Area Descriptor Word 7—Chroma Key Minimum

Table 10-36. Area Descriptor Word 7—Chroma Key Minimum field descriptions

Field	Description
CKMIN_B	Chroma key minimum blue component value.
CKMIN_G	Chroma key minimum green component value.
CKMIN_R	Chroma key minimum red component value.

10.4.2.9 Area Descriptor Word 8–Next AD

Figure 10-42 shows the fields of AD Word 8, which defines the next AD address. If more than one area is to be displayed in the same plane, the next AD points to the next area descriptor. If this is the last AD in the current frame, the next AD address must be set to 0x0000_0000. The next AD address must be aligned to an 8-byte boundary (that is, the lowest three bits should be 000). Table 10-37 describes the next area descriptor address fields.

Address: Base + 0xE8 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NEXT_AD_ADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NEXT_AD_ADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-42. Area Descriptor Word 8—Next AD

Table 10-37. Area Descriptor Word 8—Next AD field descriptions

Field	Description
NEXT_AD_ADDR	The next Area Descriptor address pointer. It is used to point to the next AD in memory when more than one area is to be displayed in a plane. This field must be set to 0x0000_0000 if this is the last AD in the current frame. This address must be 64-bit boundary aligned (set the lowest 3 bits to 0).

10.4.3 Pixel Structure

The DIU has been designed for maximum flexibility in the format of its input data. Each pixel can contain as many as four basic components: alpha, red, green, and blue. The contents of several registers in an area descriptor (see Figure 10-34) determine how the DIU parses the bitmap data into the pixel components.

The first step is to define the size of each pixel (PIXEL_S+1) and the number of bits for each component of a pixel (COMP_0 to COMP_3) in the data stream. The next step is to assign the components of the pixel (alpha, R, G, and B) to the components of the data stream (COMP_0 to COMP_3). The RED_C, GREEN_C, BLUE_C, and ALPHA_C fields in the pixel format (word 0) of the AD are used to control the assignments, selecting which component is red, which is green, and so on. The three color components (red, green, and blue) must be mapped to one of the components of the data stream; alpha can be assigned to a fourth component of the data stream or the pixel can use the global alpha value specified for the current area descriptor (G_ALPHA in word 2).

The pixel format (word 0) of the AD also contains the BYTE_F bit, which can have a significant impact on the way the DIU interprets the pixel data. If the BYTE_F bit in the AD is set (that is, flipping is disabled) then the pixel is constructed as shown in Figure 10-43.

Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	COMP_3								COMP_2					COMP_1		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_1			COMP_0					0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10-43. Example of 24-bit ARGB 8:5:6:5 Pixel Structure Definition when BYTE_F = 1

Table 10-38. Example of 24-bit ARGB 8:5:6:5 Pixel Structure Definition when BYTE_F = 1

Field	Description
COMP_3	Alpha (8 bits).
COMP_2	Red (5 bits).
COMP_1	Green (6 bits).
COMP_0	Blue (5 bits).

If the BYTE_F bit in the AD is cleared (that is, bytes in the pixel are flipped) then the pixel is constructed as shown in Figure 10-44. The DIU performs the byte flipping on each pixel before it performs the mapping.

Access: User read-only

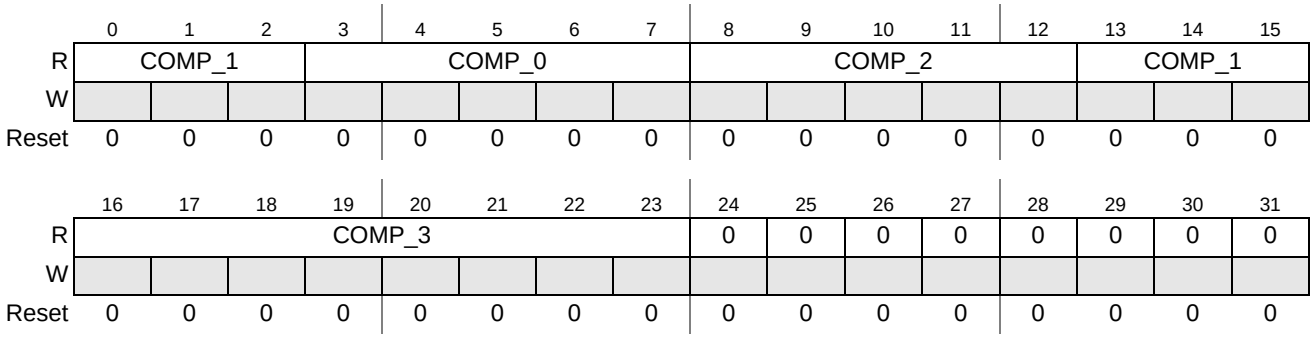


Figure 10-44. Example of 24-bit ARGB 8:5:6:5 Pixel Structure Definition when BYTE_F = 0

Table 10-39. Example of 24-bit ARGB 8:5:6:5 Pixel Structure Definition when BYTE_F = 0

Field	Description
COMP_1	Green_low (3 bits).
COMP_0	Red (5 bits).
COMP_2	Blue (5 bits).
COMP_1	Green_high (3 bits).
COMP_3	Alpha (8 bits).

10.4.4 Pixel Format Conversion

The DIU is designed to support a variety of pixel bit formats. Table 10-40 lists some examples of DIU supported pixel formats.

Table 10-40. Examples of DIU Supported Pixel Formats

Component Order 3, 2, 1, 0	Number of Bits per Component	G.Alpha	Palette
A:R:G:B	8:8:8:8	No	No
R:G:B	8:8:8	Yes	No
A:R:G:B	8:5:6:5	No	No
R:G:B	5:6:5	Yes	No
A:R:G:B	4:4:4:4	No	No
A:R:G:B	1:5:5:5	No	No
n/a	n/a	no	Yes
R:G:B:A	8:8:8:8	No	No
R:G:B:A	5:6:5:8	No	No
R:G:B:A	4:4:4:4	No	No

Table 10-40. Examples of DIU Supported Pixel Formats

Component Order 3, 2, 1, 0	Number of Bits per Component	G.Alpha	Palette
R:G:B:A	5:5:5:1	No	No
n/a	8	Yes	No

Internally, all the calculations are performed using pixels represented by 8 bits per component. For those pixel formats in which fewer than 8 bits per color components are specified, the specified bits are used as MSB, and the LSB are filled with 0s. The alpha values are extended to 8 bits as well, but the LSB are filled by sign extension to make sure that minimum value is 0 and the maximum value is extended to 255.

If the original data bitmap does not include alpha, it can be specified in the global alpha (G_ALPHA) field of the area descriptor and/or use chroma keying.

10.4.4.1 Palette Mode

As an alternate to the larger bitmap formats, the DIU supports an 8-bit pixel format through a palette table (256 × 32-bit look up table). The palette table maps an 8-bit input pixel to a 32-bit color format. The palette table is stored in a 256 × 32 bit embedded SRAM. It is accessed by the hardware as an indexed matrix such that palettized_pixel [31:0] = palette_table[input_pixel [7:0]].

The palette table should be created in advance in memory (DDR DRAM) as a consecutive sequence of 256 × 64 bits (with bits 32 to 63 filled with 0s). Writing to the PALETTE register (described in Section 10.3.3.5, “Palette Table Pointer Register (PALETTE)”) causes the DIU to automatically reload a new palette table from memory before it starts processing the next frame. Figure 10-45 shows the format of the palette table in main memory.

Palette Entry	PALETTE Offset	Local Address [2:0]							
		000	001	010	011	100	101	110	111
0	0x000	P0_B	P0_G	P0_R	P0_A	0x00	0x00	0x00	0x00
1	0x008	P1_B	P1_G	P1_R	P1_A	0x00	0x00	0x00	0x00
2	0x010	P2_B	P2_G	P2_R	P2_A	0x00	0x00	0x00	0x00
3	0x018	P3_B	P3_G	P3_R	P3_A	0x00	0x00	0x00	0x00
.....									
254	0x7F0	P254_B	P254_G	P254_R	P254_A	0x00	0x00	0x00	0x00
255	0x7F8	P255_B	P255_G	P255_R	P255_A	0x00	0x00	0x00	0x00

Figure 10-45. Palette Table Format in Memory

10.4.5 Alpha Blending

For each pixel, besides the three color components (R, G, B), a fourth component defines the transparency of the pixel. This transparency value is called alpha and has a range between 0 (pixel is completely

transparent) to 255 (pixel is completely opaque). The following equation represents the transfer function of the blending operation including the alpha values. No alpha component is defined for plane 1, because it has no planes behind it.

Eqn. 10-1

$$\text{out\_pixel} = \frac{\text{pixel\_1} \times (255 - \alpha_2) \times (255 - \alpha_3) + \text{pixel\_2} \times \alpha_2 \times (255 - \alpha_3) + \text{pixel\_3} \times 255 \times \alpha_3}{255^2}$$

For mode 2, when only planes 2 and 3 are blended to write back, the equation changes to:

Eqn. 10-2

$$\text{writeback} = \frac{\text{plane2} \times (255 - \alpha_3) + \text{plane3} \times \alpha_3}{255}$$

10.4.6 Chroma Keying

For each area (see Section 10.4.1, “Area Descriptor”), the user needs to specify a maximum and a minimum value for the chroma keying function. The maximum and minimum values are specified as R, G, and B values to which every pixel in the bitmap is compared. If all the components are greater or equal to the minimum and less than or equal to the maximum, then the alpha component for that particular pixel is replaced with 0. The chroma keying operation is performed after the color format conversion with all the components extended to 8 bits.

To turn the chroma keying off, set the minimum to 255 and the max to 0 for each component. To produce a green-screen type of effect in which all green pixels (0,255,0) are to be turned transparent, the maximum and minimum should both be set to (0,255,0).

10.4.7 Gamma Correction

The gamma table allows the user to define a completely arbitrary transfer function at the output of each color component. The gamma table should be created in memory as a consecutive sequence of 256 bytes for each color component accessed by the hardware as an indexed matrix so that output_color_component = gamma_table[input_color_component].

All three gamma tables must be stored in memory (DDR DRAM) consecutively beginning with red, then green, and blue at the end.

Similar to the palette, the gamma table is also automatically loaded by the DIU from memory into an embedded SRAM before it starts on processing the next frame if the CPU writes to the GAMMA register (see Section 10.3.3.4, “Gamma Table Pointer Register (GAMMA)”).

The size of the SRAM is 3 × 256 bytes, but it is organized as 96 × 64 bits. So the address range for each color component table is:

- Gamma_red: 0x0~0x0FF
- Gamma_green: 0x100~0x1FF
- Gamma_blue: 0x200~0x2FF

Figure 10-46 shows the format of the gamma table in main memory.

Color Component	GAMMA Offset	Local Address [2:0]							
		000	001	010	011	100	101	110	111
Red	0x000	$G0_{red}$	$G1_{red}$	$G2_{red}$	$G3_{red}$	$G4_{red}$	$G5_{red}$	$G6_{red}$	$G7_{red}$
	0x008	$G8_{red}$	$G9_{red}$	$G10_{red}$	$G11_{red}$	$G12_{red}$	$G13_{red}$	$G14_{red}$	$G15_{red}$
	.....								
	0x0F0	$G240_{red}$	$G241_{red}$	$G242_{red}$	$G243_{red}$	$G244_{red}$	$G245_{red}$	$G246_{red}$	$G247_{red}$
	0x0F8	$G248_{red}$	$G249_{red}$	$G250_{red}$	$G251_{red}$	$G252_{red}$	$G253_{red}$	$G254_{red}$	$G255_{red}$
Green	0x100	$G0_{green}$	$G1_{green}$	$G2_{green}$	$G3_{green}$	$G4_{green}$	$G5_{green}$	$G6_{green}$	$G7_{green}$
	0x108	$G8_{green}$	$G9_{green}$	$G10_{green}$	$G11_{green}$	$G12_{green}$	$G13_{green}$	$G14_{green}$	$G15_{green}$
	.....								
	0x1F0	$G240_{green}$	$G241_{green}$	$G242_{green}$	$G243_{green}$	$G244_{green}$	$G245_{green}$	$G246_{green}$	$G247_{green}$
	0x1F8	$G248_{green}$	$G249_{green}$	$G250_{green}$	$G251_{green}$	$G252_{green}$	$G253_{green}$	$G254_{green}$	$G255_{green}$
Blue	0x200	$G0_{blue}$	$G1_{blue}$	$G2_{blue}$	$G3_{blue}$	$G4_{blue}$	$G5_{blue}$	$G6_{blue}$	$G7_{blue}$
	0x208	$G8_{blue}$	$G9_{blue}$	$G10_{blue}$	$G11_{blue}$	$G12_{blue}$	$G13_{blue}$	$G14_{blue}$	$G15_{blue}$
	.....								
	0x2F0	$G240_{blue}$	$G241_{blue}$	$G242_{blue}$	$G243_{blue}$	$G244_{blue}$	$G245_{blue}$	$G246_{blue}$	$G247_{blue}$
	0x2F8	$G248_{blue}$	$G249_{blue}$	$G250_{blue}$	$G251_{blue}$	$G252_{blue}$	$G253_{blue}$	$G254_{blue}$	$G255_{blue}$

Figure 10-46. Gamma Table Format in Memory

10.4.8 Cursor

The DIU supports a 32 × 32 hardware cursor that is overlapped on top of all three planes. Figure 10-47 shows the structure of the cursor bitmap in memory (DDR DRAM). In Figure 10-48, each cursor pixel is labeled as C(*n*,*m*) where *n* indicates the row and *m* indicates the column of the 32 × 32 matrix. The cursor bitmap data is stored in memory as a continuous sequence of 16-bit pixels, starting from the top left corner, C(0,0), and continuing across the row (left to right) and then on to the next column (top to bottom) until the last cursor pixel, C(31,31), is at the bottom right corner.

Cursor Pixel Row	CURSOR Offset	Local Address [2:0]							
		000	001	010	011	100	101	110	111
0	0x000	C(0,0)		C(0,1)		C(0,2)		C(0,3)	
	0x008	C(0,4)		C(0,5)		C(0,6)		C(0,7)	
	.....								
	0x038	C(0,28)		C(0,29)		C(0,30)		C(0,31)	
1	0x040	C(1,0)		C(1,1)		C(1,2)		C(1,3)	
	0x048	C(1,4)		C(1,5)		C(1,6)		C(1,7)	
	.....								
	0x078	C(1,28)		C(1,29)		C(1,30)		C(1,31)	
2	0x080	C(2,0)		C(2,1)		C(2,2)		C(2,3)	
	0x088	C(2,4)		C(2,5)		C(2,6)		C(2,7)	
	.....								
	0x0B8	C(2,28)		C(2,29)		C(2,30)		C(2,31)	
.....									
31	0x7C0	C(31,0)		C(31,1)		C(31,2)		C(31,3)	
	.....								
	0x7F0	C(31,24)		C(31,25)		C(31,26)		C(31,27)	
	0x7F8	C(31,28)		C(31,29)		C(31,30)		C(31,31)	

Figure 10-47. Cursor Structure in Memory

Each cursor pixel is in 16-bit, ARGB 1:5:5:5 format, as shown in Figure 10-48. The field descriptions for a cursor pixel are provided in Table 10-41.

NOTE

The cursor pixel uses little-endian byte ordering. Do a 16-bit halfword endian swap for each pixel while rendering the cursor bitmap in software.

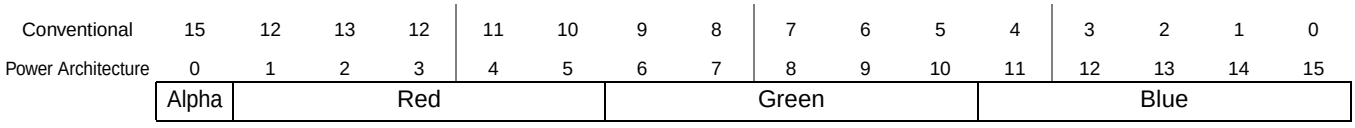


Figure 10-48. Cursor Pixel Format

Table 10-41. Field Descriptions for Cursor Pixel

Field	Description
Alpha	Alpha component of cursor pixel C(*n*,*m*).
Red	Red component of cursor pixel C(*n*,*m*).
Green	Green component of the cursor pixel is made up of $Green_{High}||Green_{Low}$.
Blue	Blue component of cursor pixel C(*n*,*m*).

To achieve cursors with shapes other than the basic square, set the alpha value to 0 in those pixels that are to be transparent. To make the cursor disappear, set the alpha value to 0 for all the pixels.

Similar to the palette and gamma, the cursor bitmap is also automatically loaded by the DIU from memory into an embedded SRAM before it starts processing the next frame if the CPU writes to the CURSOR register (See Section 10.3.3.6, “Cursor Bitmap Table Pointer Register (CURSOR)”). The SRAM size is 256 × 64 bits.

The cursor position is determined by the register CURSOR_POS (See Section 10.3.3.7, “Cursor Position Register (CUR_POS)”); the specified coordinate corresponds to the top left corner of the cursor bitmap. The cursor can be partially or totally off the screen.

10.4.9 Writeback Operation

Besides driving the LCD display, the DIU supports two operating modes (mode 2 and 3) with memory writeback operation. Intermediate blending results are stored back into the memory (DDR DRAM). These intermediate results can be then forwarded to other processing or display units or blended with other image source(s) again in the DIU, virtually extending the number of graphics planes.

DIU mode 1 supports the blending and display of as many as three planes. If an application requires the display of an image blended from four or more planes, this can be accomplished in DIU mode 2. Plane 1 is used to display one frame while the following frame is blended by the DIU. Planes 2 and 3 are used initially to read in the first and second planes to be combined, and the DIU writes back a combined plane 1 + 2. Then, while the DIU displays the same frame on plane 1, the DIU reads back in on plane 2 the intermediate plane 1 + 2, and on plane 3 the new third plane, writing back a combined plane 1 + 2 + 3. Then (while the DIU displays the same frame on plane 1), the DIU reads back in on plane 2 the intermediate plane 1 + 2 + 3, and on plane 3 the new fourth plane, writing back the combined plane 1 + 2 + 3 + 4. This combined plane is then ready to be displayed on plane 1 in the following frame, or extra steps can be added to blend more than four planes.

The writeback pixel format is always 24-bit RGB 8:8:8. stored in the memory continuously and packed.Table 10-42 shows the format of the writeback pixels in main memory.

Table 10-42. writeback Pixel Format in Memory

Memory Offset	Local Address [2:0]							
	000	001	010	011	100	101	110	111
0x00	B0	G0	R0	B1	G1	R1	B2	G2
0x08	R2	B3	G3	R3	B4	G4	R4	B5
0x10	G5	R5	B6	G6	R6	B7	G7	R7
0x18	B8	G8	R8	B9	G9	R9	B10	G10
0x20	R10	...	...	...	...	...	...	...

The register WB_MEM_ADDR (described in Section 10.3.3.13, “Writeback Plane Address Register (WB_MEM_ADDR)”) specifies the address in the memory to which the writeback data should be written. Writing to this address triggers a writeback frame refresh. When enabled, a VSYNC_WB interrupt occurs at the end of a writeback frame.

The writeback frame size is specified by the WB_SIZE register (see Section 10.3.3.12, “Writeback Plane Size Register (WB_SIZE)”).

NOTE

In mode 2, the DIU works on the basis that the writeback frame completes before the end of the display frame parallel to it (i.e. before the vertical front porch). A WB_PEND interrupt can be generated (if enabled), indicating the writeback operation did not complete in time.

10.4.10 Color Bar Generation

For testing purposes, it is desirable to generate color bars within the DIU itself. This is achieved by setting the DIU_MODE register to 4. The pattern produced is a simple one with eight vertical color bars. This mode allows the user to verify that the DIU is operational without the need to interact with the system memory. A basic color sequence is preloaded, but the user can overwrite the default values if needed. See Section 10.3.3.20, “COLBAR Registers,” for more information.

The size of the bars is set by dividing the horizontal resolution by 8 using integer math. If the horizontal resolution is not divisible by 8 exactly, not all color bars have the same width.

10.4.11 Interrupt Generation

The DIU generates interrupts through a single line controlled by the contents of two registers: INT_STATUS (see Section 10.3.3.18, “Interrupt Status Register (INT_STATUS)”) and INT_MASK (see Section 10.3.3.19, “Interrupt Mask Register (INT_MASK)”). When an interrupt occurs, the host needs to read the INT_STATUS register to find the source of the interrupt. This read operation clears the register.

Six interrupt statuses are defined:

- VSYNC—Vertical Synchronization
- VSYNC_WB—Vertical Synchronization for Writeback

- UNDRUN—Under Run Exception
- PARERR—Display Parameter Error
- LS_BF_VS—Lines Before VSYNC
- WB_PEND—Writeback Pending

A display parameter error interrupt is generated if the user sets the display parameters incorrectly. Table 10-43 lists the parameter error conditions under different DIU operating modes. When a PARERR interrupt is detected, the user can turn off the DIU (program the DIU_MODE to 0), correct the wrong parameter(s), and turn it on again. The DIU is initialized internally.

Table 10-43. Parameter Error Conditions

Mode 1 & 2	DELTA_XI > DELTA_XS
	DELTA_YI > DELTA_YS
	DELTA_XI + OFFSET_XD > DELTA_X
	DELTA_YI + OFFSET_YD > DELTA_Y
	DELTA_XI + OFFSET_XI > DELTA_XS
	DELTA_YI + OFFSET_YI > DELTA_YS
Mode 2 & 3	DELTA_XI > DELTA_X_WB
	DELTA_YI > DELTA_Y_WB
	DELTA_XI + OFFSET_XD > DELTA_X_WB
	DELTA_YI + OFFSET_YD > DELTA_Y_WB
	DELTA_XI + OFFSET_XI > DELTA_XS
	DELTA_YI + OFFSET_YI > DELTA_YS

10.4.12 Dynamic Priority Generation

The multi-port DRAM controller has built-in arbiters that use a 4-bit priority signal (i.e., 16 levels of priority). The DRAM controller tries to service the request with the highest priority first. The task of setting the priorities is done by the DRAM Controller Priority Manager block, the priority manager. It dynamically sets the priorities of the DRAM buses in such a way that bandwidth is divided fairly and each master receives its fair share of the bandwidth. Programming the look-up tables in the priority manager allows controlling relative priority to other channels and the average share of the bandwidth the current master gets.

The DIU outputs a 4-bit priority signal to the priority manager. The priority is a function of the internal input buffer filling level (buffer full → low priority, buffer empty → high priority). The buffer filling level ranges from 0 to 7 and its used to select a LUT component from the PLUT register as the priority. Filling level 0 selects PRIORITY_0 and filling level 7 selects PRIORITY_7.

The priority manager can be programmed to take the DIU priority output directly (option 1) or to follow the normal priority schema (option 2). It is recommended to use option 2, the default option.

To use option 1, the user should set PRIOMAN_CONFIG2[DIU-OVERRULE] (refer to DRAM Controller Priority Manager chapter), and program a set of priority values in the PLUT register

(PRIORITY_0 to PRIORITY_7 from high to low). The priority value ranges from 0 to 15 and accesses are blocked if the priority value is 0.

To use option 2, it is recommended to program PRIOMAN_CONFIG2[LUT SEL0] to 3, so that the priority manager selects the alternate look-up table for the DIU if DIU incoming priority bit 3 is high. So the user can set the DIU priority low in the main look-up table and set it high in the alternate look-up table, and program the PLUT register so for example its priority bit 3 is high (0x8) when the buffer is lower than half full. The purpose to do this is to only escalate the DIU priority when necessary, and save the DRAM bandwidth to the other masters like the Power Architecture e300 core.

10.4.13 Display Signal Timing

The first step to generate appropriate timing signals for the selected display is to adjust the frequency of the pixel clock to a frequency within the specified parameters of the display (see Section 10.4.13.1, “Refresh Rate”). Program the SCFR1 register in the system clock module, SCFR1[DIU_DIV], to set the divide ratio for the DIU pixel clock (refer to the system clock chapter).

Then the horizontal and vertical synchronize signals are generated based on the pixel clock. The relationship between pixel clock, HSYNC, and VSYNC signals is shown in diagram Figure 10-49 and Figure 10-50.

Refer to Section 10.3.3.11, “Display Size Register (DISP_SIZE),” Section 10.3.3.14, “Horizontal Sync Pulse Parameters Register (HSYN_PARA),” Section 10.3.3.15, “Vertical Sync Pulse Parameters Register (VSYN_PARA),” and Section 10.3.3.16, “Synchronization Signals Polarity Register (SYN_POL),” for related display parameter configuration registers.

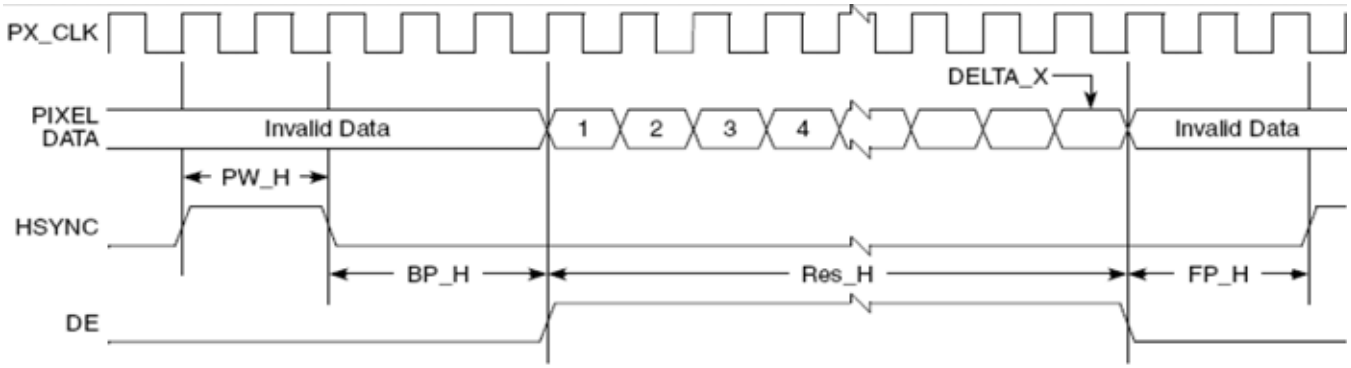


Figure 10-49. Horizontal Sync Signals

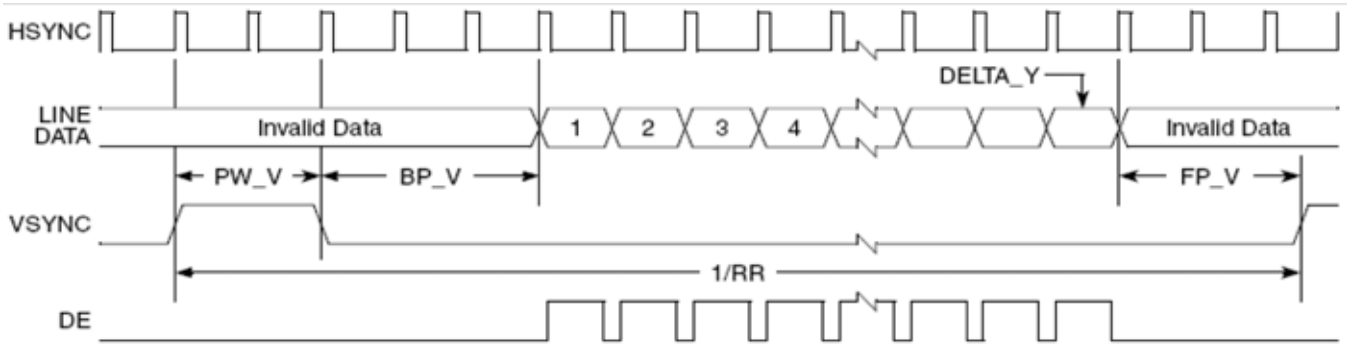


Figure 10-50. Vertical Sync Signals

By default, the DIU outputs (data and sync signals) switch at same time with the pixel clock rising edge. To provide flexibility in meeting the timing requirements of different LCD display drivers, the user can perform minor tuning of the timing of the pixel clock found on the DIU_CLK pin relative to all the other DIU signals (DIU_LD[23:0], DIU_VSYNC, DIU_HSYNC, DIU_DE). This phase tuning is done using programmable parameters in the DCCR register, specifically DCCR[CLK_INV] and DCCR[DLY_NUM] (refer to Chapter 5, "Clocks and Low-Power Modes"). Phase tuning using CLK_INV and DLY_NUM does not change the pixel clock frequency or duty cycle.

10.4.13.1 Refresh Rate

The refresh rate (or frame rate) is the number of times that the display is updated in a second. It can be calculated from the timing parameters using this formula:

Eqn. 10-3

$$rr = \frac{pix\_clk}{(delta\_x + fp\_h + pw\_h + bp\_h) \times (delta\_y + fp\_v + pw\_v + bp\_v)}$$

Because the user probably has a set target refresh rate (RR), the PIX_CLK value has been set already and the DELTA_X and DELTA_Y values are determined exactly by the panel used. The rest of the parameters in this equation must be chosen to approach the desired refresh rate while complying with the requirements established in the panel's data sheet for the front and back porches.

10.5 Initialization/Application Information

10.5.1 DIU Initialization

The procedure to bring up the DIU out of hardware reset state, start executing data processing, and display functions is as follows:

1. Hardware reset.
2. Configure I/O function multiplexing and drive strength for DIU related pins.
3. Program the display timing signal generation related registers. Failing to set appropriate values for the display timing parameters may result in damage to the display.

4. Program DIU pixel clock divide ratio and turn on the clock in the system clock module. Enable pixel clock inversion or the programmable pixel clock delay if necessary so that the interface AC timing requirement can be met.
5. Program the DIU PLUT register and the DRAM controller priority manager as appropriate so that the DIU priority can be escalated dynamically, to make sure no buffer underrun would be hit.
6. Prepare the palette, gamma tables, and cursor bitmap in memory (DDR DRAM) and set the pointers to them. This step is not strictly required because the user might be using it in an application without palette, gamma, or cursor. However, it is recommended to load zero contents in this case so that the internal SRAM can be initialized. The gamma table is always in use except in mode 3.
7. Prepare the area descriptors in memory (DDR DRAM) and set the pointers to them. This step is also not strictly required because the default background colors can be displayed or a mode that does not require area descriptors (mode 4) may be selected.
8. Program the INT_MASK register and enable the interrupts needed for the application (by default, all interrupts are disabled after hardware reset).
9. Configure the WB_MEM_ADDR register if the operating mode is mode 2 or 3.
10. Set the operating mode (by configuring the DIU_MODE register) to turn on the DIU.

10.5.2 Controlling DIU Planes after the DIU is Enabled

The DIU is initialized by correctly configuring its registers before enabling the DIU by setting the DIU_MODE to a legal, non-zero value. The DIU supports as many as three planes, which are activated by setting the corresponding DESC_*n* register to a non-zero value, and deactivated by setting the corresponding DESC_*n* register to 0x0000_0000.

10.5.3 Synchronizing with the Host (CPU)

Three interrupt status bits are defined in the DIU for synchronization purpose. They are VSYNC, LS_BF_VS, and VSYNC_WB.

If enabled, the VSYNC status bit is always asserted in the first cycle in which the VSYNC pulse is active. With this interrupt the host can always observe the beginning of a new frame. Beside this, another interrupt, LS_BF_VS (lines before vsync) can be used to set a deadline for the host to program the DIU for the next frame. This interrupt is asserted at a user-specified number of lines (set by the LS_BF_VS threshold, Figure 10-17) before the vertical front porch (FP_V).

Figure 10-51 shows a timing diagram on how these two interrupts can be used to synchronize the host and the DIU. At the end of each frame the DIU starts processing the next frame by first loading the necessary AD, palette, cursor, and gamma information from external memory pointed to by its internal register values. All of this takes place in the time window marked by the two dotted blue lines in Figure 10-51. The host needs to make sure the proper data is in external memory and proper address values are programmed into the DIU's registers before this window. Reprogramming the palette, gamma, cursor, or AD pointers in this time window is blocked. For this reason the host should use the LS_BF_VS interrupt to start setting up the DIU for the next frame, while the LS_BF_VS threshold should be set to trigger the interrupt before

FP_V pulse of the current frame (set it to greater than 0) or the host does not have enough time to properly reprogram the DIU. Use the VSYNC_WB interrupt for mode 3.

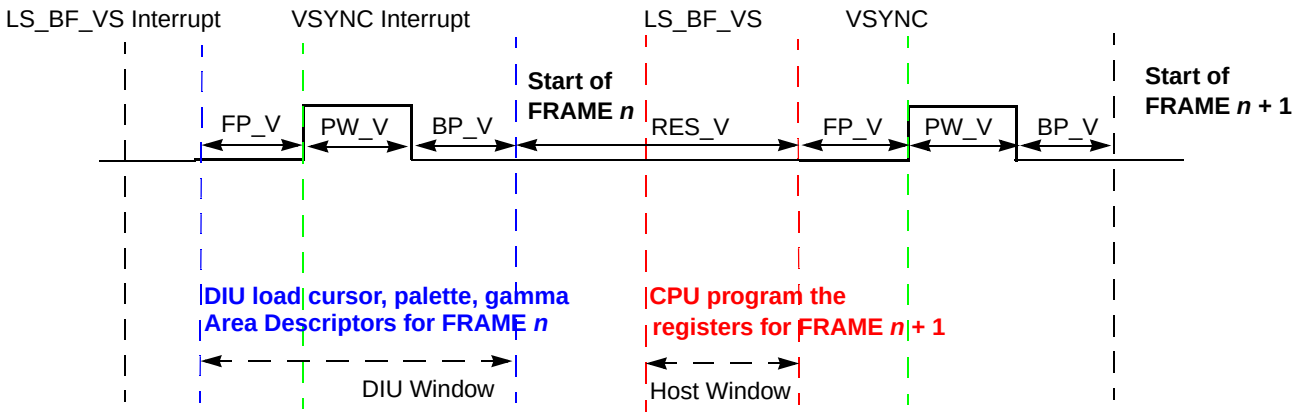


Figure 10-51. Synchronizing the Host and the DIU

For operating mode with writeback operation, a VSYNC_WB interrupt can be used to find the end of a writeback frame. To enable the DIU to work on the next writeback frame, the user need to reprogram the DESC_1/DESC_2/DESC_3 pointers for related planes, and the WB_MEM_ADDR register after the VSYNC_WB interrupt is detected.

10.5.4 Recovering from Parameter Error

A parameter error exception occurs under the conditions detailed in Table 10-43. If enabled, a PARERR interrupt is issued to the host. On reception of a PARERR interrupt, the user should turn off the DIU, correct the wrong parameters, and then turn it on again (by programming the DIU_MODE register). The DIU is initialized internally.

10.5.5 Recovering from Underrun Error

In a heavily loaded system, the DIU may experience buffer underrun errors because of long DRAM access latency. To eliminate buffer underrun issues, the user should program the MPC5125 DRAM controller priority manager so that the DIU has higher priority compared to the other masters, or program the priority manager and the DIU PLUT register to enable dynamic DIU priority escalation so that the DIU priority can be high enough while its buffer filling is low.

If a single buffer underrun occurs and it is short, the DIU may repeat the pixel before the underrun and then recover automatically when the underrun is gone to minimize impact to the display. Slight underrun error(s) do not propagate between frames, but if underrun errors take place frequently, the user should turn off the DIU, escalate the DIU priority (if necessary), and then turn it on again so that it can display normally.

Chapter 11
DRAM Controller

11.1 Introduction

The DDR DRAM controller is a multi-port DRAM controller (four ports). It supports Mobile-DDR[1], DDR-1, DDR-2, and SDR memories.

A block diagram of the multi-port DRAM controller is given in Figure 11-1.

11.1.1 Overview

The DRAM controller is a multi-port controller that listens to incoming requests on the four incoming buses and decides on each rising clock edge what command needs to be sent to the DRAM.

Each incoming bus is a 64-bit bus. The four incoming buses are:

- Bus 0: the DIU
- Bus 1: the Power Architecture e300 core
- Bus 2: NFC nand flash controller
- Bus 3: DMA, USB, FEC

The block supports connection of two DRAM rank (two chip selects) and supports the four major classes of DRAM:

- Mobile-DDR (LPDDR)
- DDR1
- DDR2
- SDR

It supports these memories in 16-bit or 32-bit wide configurations. SDR is supported in only 32-bit wide configuration.

The DRAM controller listens to the incoming requests to the four buses in parallel and then sends commands to the DRAM from the highest priority bus at the current time, while the DRAM is ready to receive the command from this particular bus. If the DRAM is blocked because it needs to meet a timing requirement, the controller sends a command from a bus where there is no blockage.

For example, suppose bus one has an incoming request on priority four, and it hits in bank 1 and the page is not open (the bank needs a precharge+activate command before the request can be serviced). Bus two has an incoming request on priority five, it hits in bank two and the correct page is already open. In this case, the DRAM controller accepts the bus two request first. While it is reading from the appropriate bank,

1. JEDEC standard calls these LPDDR. Most DRAM vendors call them Mobile-DDR.

it issues the active+precharge command for the bus one request. Because the DRAM controller sees it cannot issue the read for the bus one request (the bank needs precharge + activate), it takes the bus two request first. Because it can issue the read, the correct page is open. During this, it issues the precharge + activate for the bus 1 request in the background. This request does not suffer from the bus two request being serviced first.

The embedded priority manager determines the relative priority of each bus, and this is used by the DRAM controller to determine which requests are more urgent.

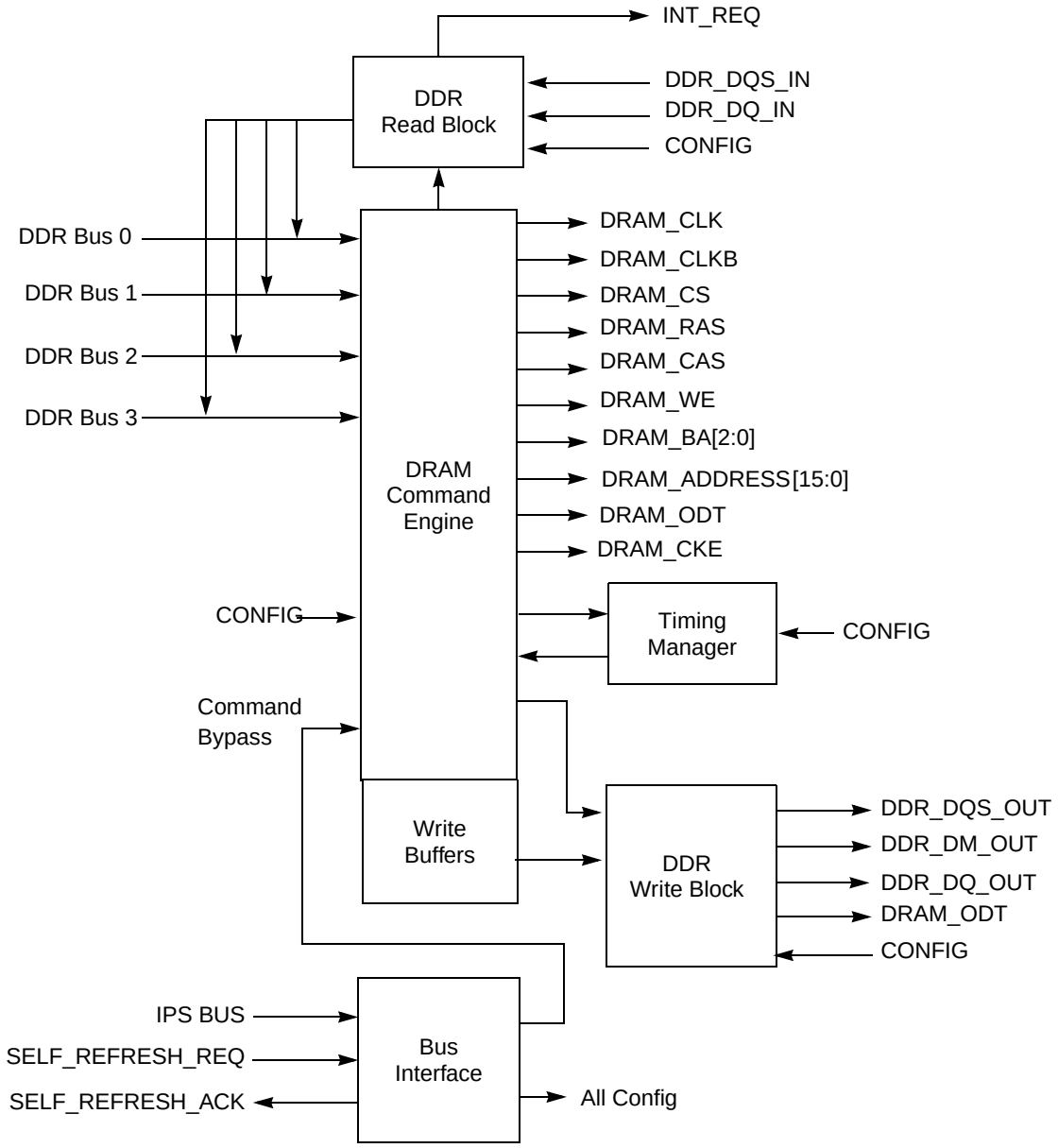


Figure 11-1. Block Diagram of the Multi-port DRAM Controller

11.2 Features

- Supports CAS latency of 2, 3, or 4 clock cycles.

- Master buses
  - Four incoming master buses
  - Supports 16-byte and 32-byte bursts
  - Supports byte enables
  - Supports 4-bit priority signal for each bus.
- Arbitration protocol
  - Inside the arbiter block, there are a total of six different arbiters that each take out the highest priority request in a certain class. All the arbiters are DRAM state aware, meaning they disregard requests that cannot be sent to the DRAM because of DRAM timing limitations.
    - Arbiter 1: Looks for highest priority read command
    - Arbiter 2: Looks for highest priority write command
    - Arbiter 3: Looks for highest priority activate-for-read command
    - Arbiter 4: Looks for highest priority activate-for-write command
    - Arbiter 5: Looks for highest priority precharge-for-read command
    - Arbiter 6: Looks for highest priority precharge-for-write command
  - After the first prioritization, the next round of arbitrating between the different arbiters is done. A fixed-priority schema is followed:
    - Read and write commands have highest priority
    - Activate has next-highest priority
    - Precharge has lowest priority
    - The DRAM is in read or write mode. In read mode, reads have priority over writes. In write mode, writes have priority over read.
    - DRAM only switches from read to write mode or vice-versa if:
      A high-priority write is found, and the write buffer is full.

      A high-priority read is found.

      The device is in read mode, but no more reads pending

      The device is in write mode, but no more writes pending.
- Write buffer contains five 32-byte entries.
- Supports 16- and 32-bit-wide DDR1/DDR2 and Mobile-DDR DRAM devices
- Supports 32-bit-wide SDR devices
- Controller supports two chip selects, 8 banks per chip select (16 banks total).
- Supports dynamic on-die termination in the host device and in the DRAM.

11.3 Memory Map and Register Definition

11.3.1 Memory Map

Table 11-1. DRAM Controller memory map

Offset from DRAM_BASE (0xFF40_9000)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x0000	DDR_SYS_CONFIG—DDR System Configuration Register	R/W	0x1000_0000	11.3.2.1/11-289
0x0004	DDR_TIME_CONFIG0—DDR Time Config0 Register	R/W	0x0000_0000	11.3.2.2.1/11-294
0x0008	DDR_TIME_CONFIG1—DDR Time Config1 Register	R/W	0x0000_0000	11.3.2.2.2/11-295
0x000C	DDR_TIME_CONFIG2—DDR Time Config2 Register	R/W	0x0000_0000	11.3.2.2.3/11-295
0x0010	DDR_COMMAND—DRAM Command Register	R/W	0x0000_0000	11.3.2.3/11-299
0x0014	DDR_COMPACT_COMMAND—Compact Command Register	R/W	0x0000_0000	11.3.2.4/11-299
0x0018	SELF_REFRESH_CMD_0—Self-Refresh Command Register 0	R/W	0x0000_0000	11.3.2.5/11-301
0x001C	SELF_REFRESH_CMD_1—Self-Refresh Command Register 1	R/W	0x0000_0000	11.3.2.5/11-301
0x0020	SELF_REFRESH_CMD_2—Self-Refresh Command Register 2	R/W	0x0000_0000	11.3.2.5/11-301
0x0024	SELF_REFRESH_CMD_3—Self-Refresh Command Register 3	R/W	0x0000_0000	11.3.2.5/11-301
0x0028	SELF_REFRESH_CMD_4—Self-Refresh Command Register 4	R/W	0x0000_0000	11.3.2.5/11-301
0x002C	SELF_REFRESH_CMD_5—Self-Refresh Command Register 5	R/W	0x0000_0000	11.3.2.5/11-301
0x0030	SELF_REFRESH_CMD_6—Self-Refresh Command Register 6	R/W	0x0000_0000	11.3.2.5/11-301
0x0034	SELF_REFRESH_CMD_7—Self-Refresh Command Register 7	R/W	0x0000_0000	11.3.2.5/11-301
0x0038	DQS_CONFIG_OFFSET_COUNT—DQS Config Offset Count Register	R/W	0x0000_0000	11.3.2.6/11-302
0x003C	DQS_CONFIG_OFFSET_TIME—DQS Config Offset Time Register	R/W	0x0000_0000	11.3.2.7/11-302
0x0040	DQS_DELAY_STATUS—DQS Delay Status Register	R	0x0000_0000	11.3.2.8/11-303
0x0044–0x005F	Reserved			
0x0060	EXTRA_ATTRIB—DDR Extra Attributes Register	R/W	0x0000_0000	11.3.2.9/11-303
0x0064–0x0FFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

3 In this column, the symbol “U” indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

11.3.2 Register Descriptions

11.3.2.1 DDR System Configuration Register (DDR_SYS_CONFIG)

Address: Base + 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RST_B	CKE	CLK ON	CMD MODE	DRAM_ROW_SELECT				DRAM_BANK_SELECT				READ_TEST		SELF REF EN	16BIT MODE	RDLY [3]
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RDLY[2:0]			HALF DQS DLY	QUART DQS DLY	WDLY[2:0]			EARLY ODT	ON DIE TERMI-NATE			FIFO OV PEND	FIFO UV PEND	FIFO OV EN	FIFO UV EN
W											FIFO OV CLEAR	FIFO UV CLEAR				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-2. DDR System Configuration Register (DDR_SYS_CONFIG)

Table 11-2. DDR_SYS_CONFIG field descriptions

Field	Description
RST_B	DRAM controller soft reset. When this bit is 0, the DRAM controller is in the reset state. When this bit is 1, the DRAM controller is out of reset. The bit controls the reset to the internal state machines. The configuration registers are reset by the resets from the hardware reset block, not by this bit.
CKE	Value on the DRAM CKE pin. For functional operation, this needs to be high. During power-down, value can be low.
CLK ON	When this bit is 1, the DRAM clock is running. When this bit is 0, the DRAM clock is stopped
CMD MODE	When this bit is 0, the DRAM controller is in normal operation. When this bit is 1, the DRAM controller is in command mode and does not respond to requests on the incoming buses. Command mode is used for DRAM initialization and to switch the DRAM into and out of the different power-down and self-refresh modes.
DRAM_ROW_SELECT and DRAM_BANK_SELECT	These fields control the multiplexing of the bus address to the DRAM bank and row address. Table 11-5 and Table 11-6 give the details. DRAM column address depends on 16-bit mode bit, and relationship is given in Table 11-5.
READ_TEST	These fields are for production test. Do not use.
SELFREFEN	Self-refresh enable. When this bit is 1, the DRAM controller autonomously enters and exits the self-refresh using the self-refresh command registers when requested by the PMC. When the bit is 0, the transition is blocked.
16-BIT MODE	When this bit is set, the DRAM controller assumes a 16-bit wide memory is used. When this bit is cleared, a 32-bit wide memory is assumed. **Note:** This does not configure the pins for 16-bit mode. That must be done in the pin configuration.

Table 11-2. DDR_SYS_CONFIG field descriptions (Continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>RDLY[3:0]</td><td>This field controls the expected delay between sending a read command to the DRAM and receiving the read data from the DRAM.
RDLY, HALF DQS DLY, and QUART DQS delay together to code for t_{DQSEN}.
The t_{DQSEN} is the delay between the read command and when the internal DQS enable goes high. See Figure 11-3. Timing is internally compensated, and is referred to timing at the device pins. t_{DQSEN} should be selected so the L-H transition of DQS enable is always in the preamble of the DQS input of the READ command. Required t_{DQSEN} value depends on the CAS latency (CL), the distance between the DRAM and the device, and the type of DRAM used. Table 11-3 gives the detail on programming t_{DQSEN}.</td></tr>
<tr><td>HALF DQS DLY</td><td>This field is an extra field to control the expected read delay between issuing the read command and getting read data from the DRAM. This field offers 1/2 CSB clock granularity when programming the delay. See description of field RDLY for details.</td></tr>
<tr><td>QUART DQS DLY</td><td>This field is an extra field to control the expected read delay between issuing the read command, and getting read data from the DRAM. This field offers 1/4 csb clock granularity when programming the delay. See the description of the RDLY bitfield.</td></tr>
<tr><td>WDLY[2:0]</td><td>This field controls the write latency (WL) for write commands.
<table>
<tr><th>WDLY[2:0]</th><th>Write Latency (CSB Clocks)</th></tr>
<tr><td>0b001</td><td>1</td></tr>
<tr><td>0b010</td><td>2</td></tr>
<tr><td>0b011</td><td>3</td></tr>
<tr><td>0b100</td><td>4</td></tr>
<tr><td colspan="2">Other values of WDLY are reserved.</td></tr>
</table>
</td></tr>
<tr><td>EARLY ODT</td><td>This bit needs to be set if write latency is 1 (WDLY[2:0] = 001) and on die termination is used with DDR2 DRAM. It makes sure the DRAM controller asserts the ODT signal going to the DRAM one clock ahead of issuing the write command.</td></tr>
<tr><td>ON DIE TERMINATE</td><td>This bit controls on-die termination (ODT) in the controller. If this bit is 1, the internal pads generate ODT during read. If the bit is 0, no ODT is provided. The ODT in the DRAM is controlled via the DRAM internal configuration registers. Please consult DRAM data sheet for it.</td></tr>
</table>

Table 11-2. DDR_SYS_CONFIG field descriptions (Continued)

Field	Description
FIFO OV CLEAR FIFO UV CLEAR FIFO OV PENDING FIFO UV PENDING FIFO OV EN FIFO UV EN	These bits control the interrupt generation by the DRAM controller. The DRAM controller has two interrupts: FIFO OV pending and FIFO UV pending. These interrupts are set on overflow or underflow of the FIFO in the read block. When a read command is sent to the DRAM, it is entered into a FIFO. The DRAM is expected to answer by sending back the read data with some up and down edges on the DQS lines (the DQS strobes) used to clock the data. The DRAM controller clocks the read data with the DQS strobes supplied by the DRAM and retrieves the read command from the FIFO after receiving the correct number of read strobes. When the read data strobes returned by the DRAM do not match the expectations of the controller, the FIFO may underflow (if too many DQS strobes are coming back from the DRAM) or overflow (if not enough DQS strobes are coming back). These underflows and overflows are basically the result of problems with the DRAM interface or incorrect parameter settings in the controller or the DRAM. Care has been taken during the design of the DRAM controller not to enter a hang-up state when this occurs. However, read data is corrupt and CPU is informed via the FIFO overflow and FIFO underflow interrupts. The issue is also discussed in Section 11.4.7, “Bus Interface.” • FIFO_OV_PENDING and FIFO_UV_PENDING signal to the CPU if an overflow or underflow interrupt is pending • FIFO_OV_EN and FIFO_UV_EN bits are interrupt enable bits. If the pending + enable bit is set at the same time, the interrupt is sent to the e300 CPU. • FIFO_OV_CLEAR and FIFO_UV_CLEAR are clear bits. Writing a 1 to either of these clears the pending interrupt.

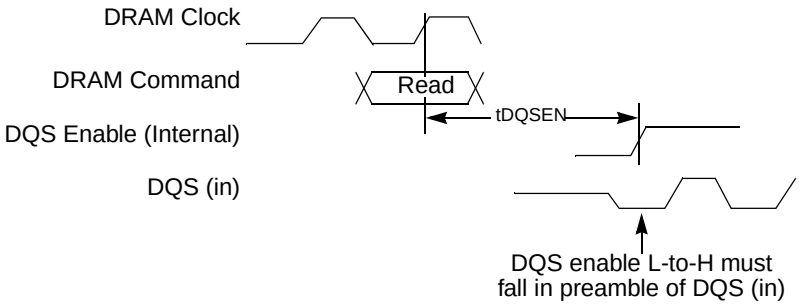


Figure 11-3. t_{DQSEN}

Table 11-3. Programming t_{DQSEN}

{rdly[3:0],half_dqs_dly,quart_dqs_dly}	t_{DQSEN} (CSB Clock Periods)
1000 0 0	0.5
1000 0 1	0.75
1000 1 0	1.0
1000 1 1	1.25
0100 0 0	1.5
0100 0 1	1.75
0100 1 0	2.0
0100 1 1	2.25
0010 0 0	2.5
0010 0 1	2.75

Table 11-3. Programming t_{DQSEN} (Continued)

{rdly[3:0],half_dqs_dly,quart_dqs_dly}	t_{DQSEN} (CSB Clock Periods)
0010 1 0	3.0
0010 1 1	3.25
0001 0 0	3.5
0001 0 1	3.75
0001 1 0	4.0
0001 1 1	4.25
0000 0 0	4.5
0000 0 1	4.75
0000 1 0	5.0
0000 1 1	5.25

Table 11-4. Number of DRAM Banks Addressed and Mapping of Address to DRAM Bank Address

DRAM_BANK_SELECT	Number of Banks	DRAM_BANK
0	4	DRAM_BANK[1:0] = address[11:10]
1	4	DRAM_BANK[1:0] = address[12:11]
2	8	DRAM_BANK[2:0] = address[13:11]
3	4	DRAM_BANK[1:0] = address[13:12]
4	8	DRAM_BANK[2:0] = address[14:12]
5	4	DRAM_BANK[1:0] = address[14:13]
6	8	DRAM_BANK[2:0] = address[15:13]
7	4	DRAM_BANK[1:0] = address[15:14]
8	8	DRAM_BANK[2:0] = address[16:14]
9	4	DRAM_BANK[1:0] = address[25:24]
10	8	DRAM_BANK[2:0] = address[26:24]
11	4	DRAM_BANK[1:0] = address[26:25]
12	8	DRAM_BANK[2:0] = address[27:25]
13	8	DRAM_BANK[2:0] = address[28:26]
14	8	DRAM_BANK[2:0] = address[29:27]
15	8	DRAM_BANK[2:0] = address[30:28]

Table 11-5. Mapping of Address to DRAM Column Address

16-Bit Mode	DRAM_COLUMN[12:0]
0	DRAM_COLUMN[12:0] = address[14:2]
1	DRAM_COLUMN[12:0] = address[13:1]

NOTE

In 16-bit mode (16BITMODE = 1), DDR memories with column address line 0 to 7 are not supported.

Table 11-6. Mapping of Address to DRAM Row Address

DRAM_ROW_SELECT [2:0]	DRAM_ROW[15:0]
0	DRAM_ROW[15:0] = address[25:10]
1	DRAM_ROW[15:0] = address[26:11]
2	DRAM_ROW[15:0] = address[27:12]
3	DRAM_ROW[15:0] = address[28:13]
4	DRAM_ROW[15:0] = address[29:14]
5	DRAM_ROW[15:0] = address[30:15]
6	DRAM_ROW[14:0] = address[30:16]
7	DRAM_ROW[13:0] = address[30:17]

Table 11-7. Mapping of Address to DRAM Chip Select

DRAM_CS_SELECT[3:0][1]	DRAM_CS
0	dram_cs_select[2] = 0
1	dram_cs_select = address[12]
2	dram_cs_select = address[13]
3	dram_cs_select = address[14]
4	dram_cs_select = address[15]
5	dram_cs_select = address[16]
6	dram_cs_select = address[27]
7	dram_cs_select = address[28]
8	dram_cs_select = address[29]
9	dram_cs_select = address[30]

1 For the field of register 0x60, see Table 11-17.

2 if DRAM_CS_SELECT = 0 → use CS0.
if DRAM_CS_SELECT = 1 → use CS1.

11.3.2.2 Timing Configuration

11.3.2.2.1 DDR Time Configuration Register 0 (DDR_TIME_CONFIG0)

Address: Base + 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DRAM_REFRESH_TIME[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DRAM_COMMAND_TIME[7:0]								DRAM_BANK_PRE_TIME[7:0]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-4. DDR Time Configuration Register 0 (DDR_TIME_CONFIG0)

Table 11-8. DDR_TIME_CONFIG0 field descriptions

Field	Description
DRAM_REFRESH_TIME [15:0]	Refresh interval of the DRAM. Program in this register the number of CSB clocks between any two refresh requests. This register should contain the maximum number of CSB clocks between two refresh requests. The average time in CSB clock periods between two refreshes to the DRAM is this number.
DRAM_COMMAND_TIME [7:0]	Time-out after sending a command to the DRAM in bypass mode. For command sent to the DRAM using the DDR_COMMAND and DDR_COMPACT_COMMAND register, the normal checking of the timing parameters is not done. Instead, any new command to the DRAM is disabled for DRAM_COMMAND_TIME[7:0] dram clock periods. This parameter needs to be programmed for the worst-case time-out.
DRAM_BANK_PRE_TIME [7:0]	Time-out. Any active bank, that has no outstanding requests, is automatically precharged by the DRAM controller after this time-out has elapsed since the last access to the bank. This time can be set short, which results in open banks being precharged quite fast to long, which results in open banks left open for a long time. The value is a time count in DRAM clock periods.

11.3.2.2.2 DDR Time Configuration Register 1 (DDR_TIME_CONFIG1)

Address: Base + 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DRAM_TIME_RFC[5:0]						DRAM_TIME_WR1[4:0]					DRAM_TIME_WTR1[3:0]				DRAM_TIME_RRD[5]
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DRAM_TIME_RRD[4:0]						DRAM_TIME_RC[5:0]					DRAM_TIME_RAS[4:0]				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-5. DDR Time Configuration Register 1 (DDR_TIME_CONFIG1)

11.3.2.2.3 DDR Time Configuration Register 2 (DDR_TIME_CONFIG2)

Address: Base + 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DRAM_TIME_RCD[3:0]				DRAM_TIME_FAW[4:0]					DRAM_TIME_RTW1[3:0]			DRAM_TIME_CCD[3:1]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DRAM_TIME_CCD[0]	DRAM_TIME_RTP[4:0]							DRAM_TIME_RP[4:0]				DRAM_TIME_RPA[4:0]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-6. DDR Time Configuration Register 2 (DDR_TIME_CONFIG2)

The DDR_TIME_CONFIG1 and DDR_TIME_CONFIG2 registers need to be programmed with the DDR1/DDR2 timing parameters. All times are given in clock cycles.

The timing parameters are conceived so the controller CSB clock cycles match with the JEDEC DDR2 specification. To interface with DDR1 or Mobile-DDR (LPDDR), some timing parameters need not be enforced, or are calculated differently. Refer to the DRAM datasheet to determine their value. The timing parameters need to be programmed in function of this DRAM requirement. Table 11-9 gives the details.

Table 11-9. Timing Parameters

Timing Parameter	Controls JEDEC Parameter (JEDEC spec)	Formulae (All times in CSB clock periods)	Description
DRAM_TIME_RFC	t_{RFC}	DRAM_TIME_RFC = t_{RFC}	REFRESH to ACTIVE or REFRESH to REFRESH command interval.

Table 11-9. Timing Parameters (Continued)

Timing Parameter	Controls JEDEC Parameter (JEDEC spec)	Formulae (All times in CSB clock periods)	Description
DRAM_TIME_RRD	t_{RRD}	DRAM_TIME_RRD = t_{RRD}	ACTIVE bank A to ACTIVE bank B command.
DRAM_TIME_RC	t_{RC}	DRAM_TIME_RC = t_{RC}	ACTIVE to ACTIVE (same bank) command.
DRAM_TIME_RAS	t_{RAS}	DRAM_TIME_RAS = t_{RAS}	ACTIVE to PRECHARGE command.
DRAM_TIME_RCD	t_{RCD}	DRAM_TIME_RCD = t_{RCD}	ACTIVE to READ or WRITE delay.
DRAM_TIME_FAW	t_{FAW}	DRAM_TIME_FAW[1] = t_{FAW}	4-bank activate period.
DRAM_TIME_CCD	t_{CCD}	DRAM_TIME_CCD[2] = max(t_{CCD},2) (32-bit mode) max(t_{CCD},4)(16-bit mode)	CAS to CAS delay Because time is needed for data to be sent over, this time is minimum two clocks in 32-bit mode and four clocks in 16-bit mode.
DRAM_TIME_RTP	t_{RTP}	DRAM_TIME_RTP[3] = t_{RTP} (32-bit mode, DDR2) t_{RTP}+2 (16-bit mode, DDR2)	Read to precharge delay. DRAM_TIME_RTP is the read-to-precharge delay and t_{RTP} is the *internal* read-to-precharge delay, hence, the difference for 16-bit mode. Figure 11-7 gives the details.
DRAM_TIME_RP	t_{RP}	DRAM_TIME_RP = t_{RP}	Precharge command period.
DRAM_TIME_RPA	t_{RP}	DRAM_TIME_RPA[4] = t_{RP} + 1 (8 bank device) DRAM_TIME_RPA = t_{RP} (4 bank device)	Precharge all command period.
DRAM_TIME_WR1	t_{WR}	DRAM_TIME_WR1 = WL + t_{WR} + 2 (32-bit mode) WL + t_{WR} + 4 (16-bit mode)	DRAM_TIME_WR1 is the write recovery time, measured in clocks between write command and precharge command. For this reason, WL (the write latency) and the length of the actual write (2 or 4) need to be added to t_{WR}. Figure 11-8 gives the details.
DRAM_TIME_WTR1	t_{WTR}	DRAM_TIME_WTR1 = WL + t_{WTR} + 2 (32-bit mode) WL + t_{WTR} + 4 (16-bit mode)	DRAM_TIME_WTR1 is the write to read time, measured in clocks between write command and read command. For this reason, WL (the write latency) and the length of the actual write (2 or 4) need to be added to t_{WTR}. Figure 11-9 gives the details.
DRAM_TIME_RTW1	—	DRAM_TIME_RTW1 = CL - WL + 2 + t_{BTA} (32-bit) CL - WL + 4 + t_{BTA} (16-bit)	DRAM_TIME_RTW1 is the read-to-write time, measured in clocks between the read and write command. There is no limitation on the DRAM on how to set this parameter. The parameter should be set such that there is no contention on the DQ data bus when switching from read to write. Equation given at left tries to come up with a formulae that defines the minimum value of DRAM_TIME_RTW1 to avoid contention. CL is the cas latency, WL is the write latency, and t_{BTA} is the bus turn-around time. t_{BTA} is the minimum dead time that needs to be put on the bus between the MPC5125 driving the bus and the DRAM driving the bus to take into account the transit delay on the PCB, the pad delay, the DRAM skew, and the on-chip delay.

Table 11-9. Timing Parameters (Continued)

Timing Parameter	Controls JEDEC Parameter (JEDEC spec)	Formulae (All times in CSB clock periods)	Description
DRAM_TIME_CCD_OTHER[5]	—	dram_time_ccd = max(t_{CCD},2) (32-bit mode) +1 max(t_{CCD},4)(16-bit mode) +1	CAS to CAS delay from one chip select to the other Because time is needed for data to be sent over, this time is minimum 2 clocks in 32-bit mode, 4 clocks in 16-bit mode
DRAM_TIME_WTR1_OTHER[5]		dram_time_wtr1 = WL - RL + 2 + 2(32-bit mode) WL - RL + 4 + 2(16-bit mode)	dram_time_wtr1 is the write to read time for write and read happening on different chip selects, *measured in clocks between write command and read command*. For this reason, WL (the write latency) and the length of the actual write (2 or 4) need to be added to t_{WTR}. Figure 11-9 gives the details.

1 For DRAMs that do not need this check, set equal to 4 × t_{RRD}

2 For DDR1 and Mobile-DDR t_{CCD} is 2 for 32-bit operation, 4 for 16-bit operation.

3 For DDR1 and Mobile-DDR mode, t_{RTP} is not explicitly given. It is equal to 4 for 16-bit mode, equal to 2 for 32-bit mode.

4 This timing parameter controls precharge all command period duration. The equations shown are the JEDEC definition of the t_{RPA}. Some DRAM vendors do not follow JEDEC on this, and list t_{RPA} directly. In this case, set *DRAM_TIME_RPA* = t_{RPA}.

5 Field is part of register 0x60, dram_extra_attributes

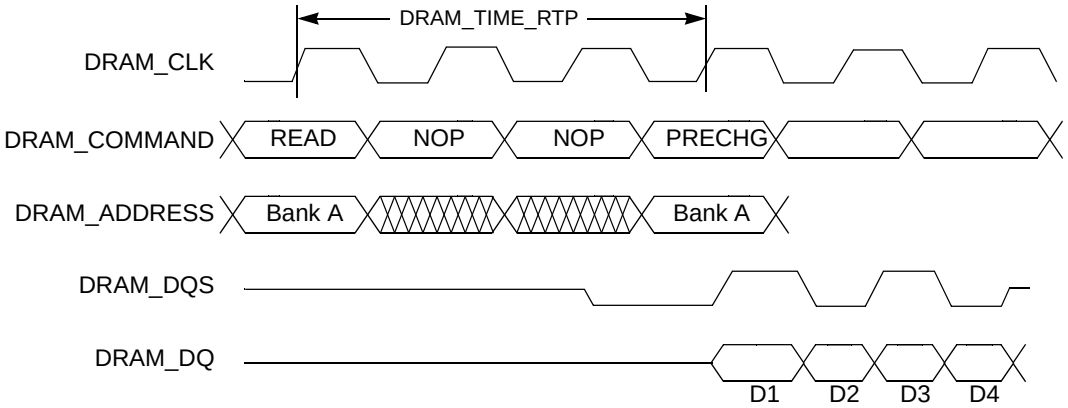


Figure 11-7. Read to Precharge Timing Diagram

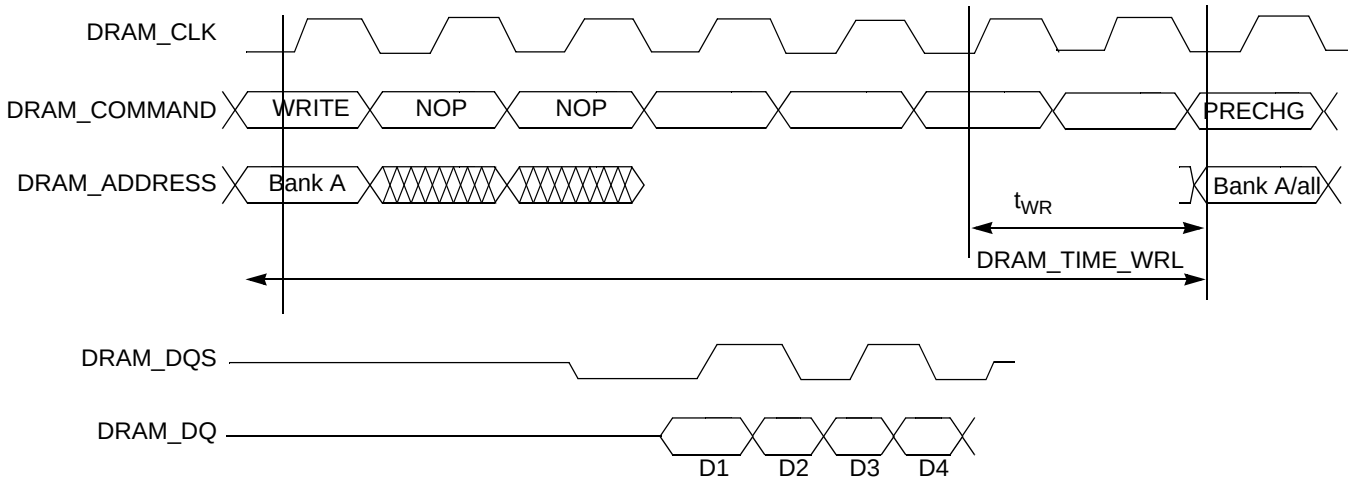


Figure 11-8. Write to Precharge Timing Diagram

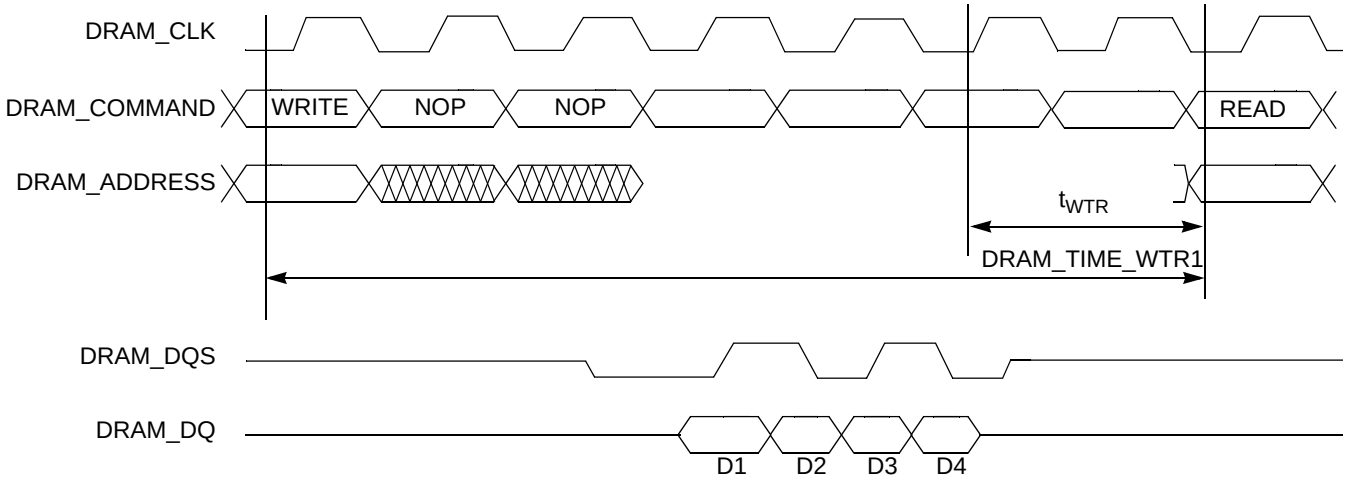


Figure 11-9. Write to Read Timing Diagram

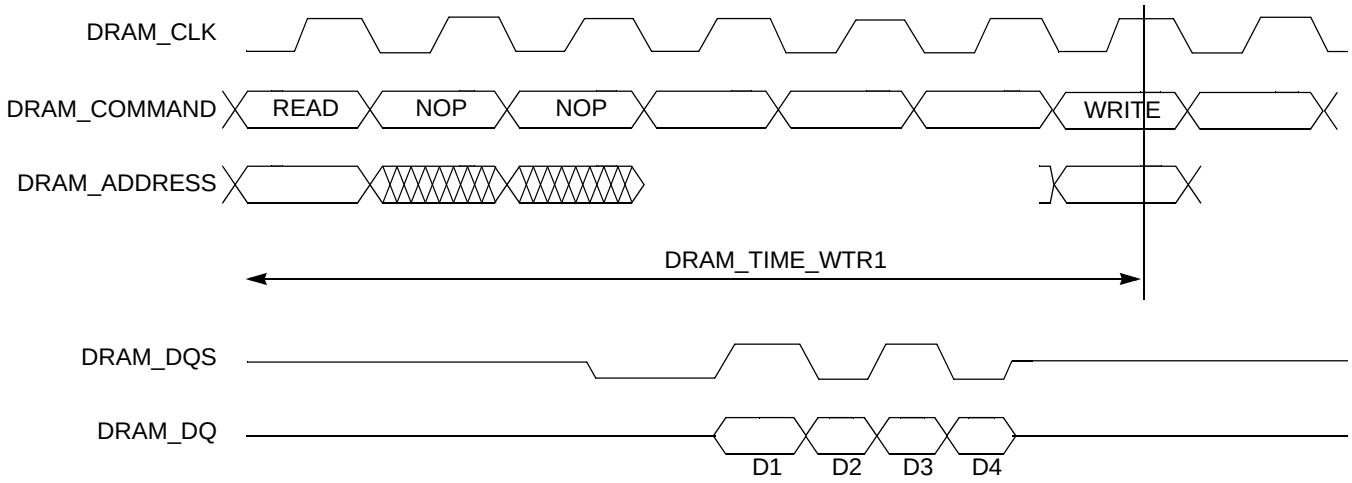


Figure 11-10. Read to Write Timing Diagram

11.3.2.3 DRAM Command (DDR_COMMAND) Register

The DRAM Command (DDR_COMMAND) register gives the option to send commands directly to the DRAM. This register only operates when the command mode bit (CMD MODE, bit 3) is set in the DDR_SYS_CONFIG register.

Address: Base + 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W									DRAM_COMMAND[23:16]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	DRAM_COMMAND[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-11. DRAM Command (DDR_COMMAND) Register

Table 11-10. DDR_COMMAND field descriptions

Field	Description
DRAM_COMMAND [23:0]	When the DDR_SYS_CONFIG[CMD MODE] bit is set, the value written to bits [23:0] of this register is output on the DRAM address group with following mapping: • DRAM_ADDRESS[14:0] = DRAM_COMMAND[14:0] • DRAM_ADDRESS[15] = 0 • if(DRAM_COMMAND[15] == 1) turn off CKE DRAM attribute bit[1] • DRAM_BA[2:0] = DRAM_COMMAND[18:16] • DRAM_WEB = DRAM_COMMAND[19] • DRAM_CAS = DRAM_COMMAND[20] • DRAM_RAS = DRAM_COMMAND[21] • DRAM_CS0 = DRAM_COMMAND[22] • DRAM_CS1 = DRAM_COMMAND[23] **Note:** The intended use of the command interface is to initialize the DRAM and to put the DRAM into or out of the self-refresh and power-down modes.

1 If CKE is turned off, it is turned off on the same clock cycle as when the command is being written to the DRAM.

11.3.2.4 Compact Command Register (DDR_COMPACT_COMMAND)

Address: Base + 0x0014 Access: User write-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	DRAM_COMPACT_COMMAND[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-12. Compact Command (DDR_COMPACT_COMMAND) Register

Table 11-11. DDR_COMPACT_COMMAND field descriptions

Field	Description
DRAM_COMPACT_COMMAND[15:0]	The compact command register gives the option to sent commands to the DRAM using 16-bit writes. See Table 11-12.

Table 11-12. DDR_COMPACT_COMMAND register options

COMPACT_COMMAND[15:14]	Description
00	Write DRAM attributes and wait (wait time = wait time till next command) Wait is executed after writing attributes. • CKE = COMPACT_COMMAND[13] • Self Ref En = COMPACT_COMMAND[12] • CLK ON = COMPACT_COMMAND[11] • CMD MODE = COMPACT_COMMAND[10] • If (COMPACT_COMMAND[7] == 1'b1) • Wait time = (COMPACT_COMMAND[6:0] × 512) dram clock periods Or • Wait time = (COMPACT_COMMAND[6:0] × 32) dram clock periods
01	DRAM command • DRAM_CS = COMPACT_COMMAND[12] • DRAM_RAS = COMPACT_COMMAND[11] • DRAM_CAS = COMPACT_COMMAND[10] • DRAM_WEB = COMPACT_COMMAND[9] • DRAM_BA[2:0] = COMPACT_COMMAND[8:6] • DRAM_ADDRESS[10] = COMPACT_COMMAND[5] • if(COMPACT_COMMAND[4] == 1'b1) turn off CKE DRAM attribute bit[1]
1x	DRAM set mode registers • DRAM_CS = 0 • DRAM_RAS = 0 • DRAM_CAS = 0 • DRAM_WEB = 0 • DRAM_ADDRESS[13] = 0 • DRAM_BA[2] = 0 • DRAM_ADDRESS[12:0] = COMPACT_COMMAND[12:0] • DRAM_ADDRESS[14:13] = COMPACT_COMMAND[14:13]

1 CKE is turned off the clock cycle when sending the requested command to the DRAM.

The COMPACT_COMMAND register's main purpose is to be written during enter/exit of self-refresh (the auto-sequencer). This is described in the following section.

The compact command register allows three types of actions to be executed:

- Write DRAM attributes and wait. Wait is executed after updating the DRAM attributes. It is possible to update the CKE bit, the self-refresh enable, the CLK configuration (on/off), and the CMD mode setting.

  If, during the time the wait is executed, another command is written to the Compact Command register, this write is delayed until the wait is over.

  During this time, the peripheral bus and all buses connected to it block and are not able to process any other read or write.

- Write a command to DRAM without controlling the address. In this mode, it is possible to send refresh, activate, and precharge commands to the DRAM
- Write a DRAM mode register.

11.3.2.5 Enter/Exit Self-Refresh Registers

Address: Base + 0x0018 (SELF_REFRESH_CMD_0) Base + 0x0028 (SELF_REFRESH_CMD_4) Access: User read/write
Base + 0x001C (SELF_REFRESH_CMD_1) Base + 0x002C (SELF_REFRESH_CMD_5)
Base + 0x0020 (SELF_REFRESH_CMD_2) Base + 0x0030 (SELF_REFRESH_CMD_6)
Base + 0x0024 (SELF_REFRESH_CMD_3) Base + 0x0034 (SELF_REFRESH_CMD_7)

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SELF_REFRESH_CMD*n*[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-13. Self-Refresh Command *n* Register

Table 11-13. SELF_REFRESH_CMD_*n* field descriptions

Field	Description
SELF_REFRESH_CMD*n*[15:0]	

The self-refresh command registers contain the commands sent to the DRAM when a self-refresh request is given. Figure 11-14 gives the details.

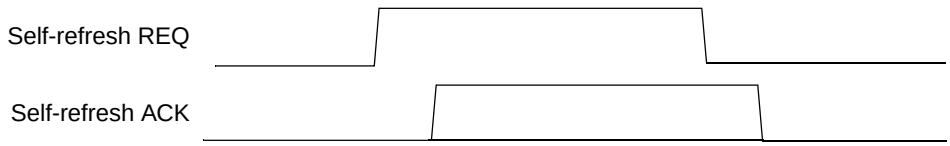


Figure 11-14. Enter/exit Self-Refresh Command Protocol

When the DRAM controller sees a low-to-high transition on the incoming self-refresh REQ signal coming from the PMC controller, its reaction depends on the state of the internal self-refresh EN command bit.

- If the self-refresh EN bit is set, the DRAM controller writes self-refresh CMD[0:3] registers, starting with reg 0 and ending with reg 3, to the compact command register. After the last register has been written and its wait time has expired (if any), it pulls high the self-refresh ACK signal to the PMC to acknowledge entry to self-refresh mode.
- If the self-refresh EN bit is clear, the DRAM controller does not react to the request and keeps self-refresh ACK signal low.

When the DRAM controller sees a high-to-low transition on the incoming self-refresh REQ signal coming from the PMC controller, its reaction is similar and depends on the state of the internal self-refresh EN command bit again.

- If the self-refresh EN bit is set, the DRAM controller writes self-refresh CMD[4:7] registers, starting with register CMD4 and ending with register CMD7, to the compact command register. After the last register has been written and its wait time has expired (if any), it pulls low the self-refresh ACK signal to the PMC to acknowledge exit from self-refresh mode.
- If the self-refresh EN bit is clear, the DRAM controller does not react to the request and keeps self-refresh ACK signal high.

11.3.2.6 DQS Config Offset Count (DQS_CONFIG_OFFSET_COUNT) Register

Address: Base + 0x0038 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	DQS SLAVE 3 OFFSET COUNT[6:0]							0	DQS SLAVE 2 OFFSET COUNT[6:0]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	DQS SLAVE 1 OFFSET COUNT[6:0]							0	DQS SLAVE 0 OFFSET COUNT[6:0]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-15. DQS Config Offset Count (DQS_CONFIG_OFFSET_COUNT) Register

Table 11-14. DQS_CONFIG_OFFSET_COUNT field descriptions

Field	Description
DQS_SLAVE_[3:0]_ OFFSET_COUNT	There is a separate field for each DQS input to the controller. These fields code for an offset counted in elemental gate delay increments applied to each DQS slave. The number is a two-complement number that can be positive and negative. This register can be used to compensate systematic delay shift in the DRAM controller due to processing. Leave this register all-zero, unless Freescale issues a report giving a different value.

11.3.2.7 DQS Config Offset Time (DQS_CONFIG_OFFSET_TIME) Register

Address: Base + 0x003C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	DQS SLAVE 3 OFFSET TIME[4:0]					0	0	0	DQS SLAVE 2 OFFSET TIME[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	DQS SLAVE 1 OFFSET TIME[4:0]					0	0	0	DQS SLAVE 0 OFFSET TIME[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-16. DQS Config Offset Time (DQS_CONFIG_OFFSET_TIME) Register

Table 11-15. DQS_CONFIG_OFFSET_TIME field descriptions

Field	Description
DQS_SLAVE_[3:0]_ OFFSET_TIME	There is a separate field for each DQS input to the controller. These fields code for an offset counted in time units. This register can be used to advance or delay the read strobe. Negative values advance the read strobe, positive values retard the read strobe. Time delay coded = [field value (2-complement)] × Tdram-clock/256. The applied offset range for a 200 MHz clock is approximately ± 290 ps.

11.3.2.8 DQS Delay Status (DQS_DELAY_STATUS) Register

Address: Base + 0x0040 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	DQS MASTER COUNT 2[11:0]											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	DQS MASTER COUNT 1[11:0]											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-17. DQS Delay Status (DQS_DELAY_STATUS) Register

Table 11-16. DQS_DELAY_STATUS field descriptions

Field	Description
DQS MASTER COUNT 2	Delay count output by the controller for the first DQS master to code for 1/4 CSB clock delay.
DQS MASTER COUNT 1	Delay count output by the controller for the second DQS master to code for 1/4 CSB clock delay.

11.3.2.9 DDR Extra Attributes (EXTRA_ATTRIB) Register

Address: Base + 0x0060 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DRAM_TIME_CCD _OTHER[3:0]				DRAM_TIME_WTR1 _OTHER				CON FIG_ SDR	CON FIG_ CAS3	CON FIG_ A15	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	DRAM_CS_SELECT[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-18. DDR Extra Attributes (EXTRA_ATTRIB) Register

Table 11-17. EXTRA_ATTRIB field descriptions

Field	Description
DRAM_TIME_CCD_OTHER[3:0]	For description, refer to Table 11-9.
DRAM_TIME_WTR1_OTHER[3:0]	For description, refer to Table 11-9.
CONFIG_SDR	0 DDR mode. 1 SDR mode.
CONFIG_CAS3	0 SDR timing with CAS latency 2. 1 SDR timing with CAS latency 3.
CONFIG_A15	0 A15 pin acts as A15. 1 A15 pin acts as Chip Select 1.
DRAM_CS_SELECT[3:0]	For description, refer to Table 11-7.

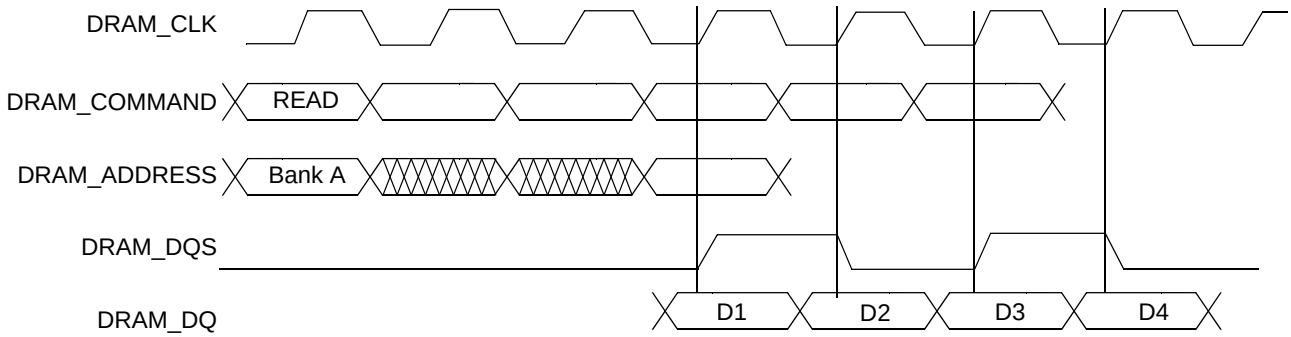


Figure 11-19. SDR read timing diagram

During SDR mode, DQS is driven by the device all the time. DQS is driven out by the controller synchronous with DRAM_CLK the specified CAS delay after the read command was issued. Read data is input relative to the DQS, exactly the same way as it happens in DDR mode. This means, the strobe point is 1/4 clock after the edges of the DQS.

It is possible to shift the sample point to right after the clock edges of DQS by writing 0x2020_2020 to the DQS Config Offset Time register.

11.4 Functional Description

The DRAM controller is a multi-port DRAM controller. It listens to incoming requests on multiple buses and decides on each rising clock edge what command needs to be sent to the DRAM.

A block diagram is given in Figure 11-1. The major blocks of the DRAM controller are described in the following sections.

11.4.1 Interfacing with the DRAM

11.4.1.1 Connecting the DRAM

- 32-bit DRAM systems need to be connected to all DQ, DM, DQS lines.

- 16-bit DRAM systems need to connected to the low order bits of the data bus. (DQ[15:0], DQS[1:0], and DM[1:0])
- Row/column address pins need to be connected starting with bit [0] and ending with the highest order DRAM bit. Leave MSB's unconnected if the DRAM has fewer address pins than the controller.
- DRAM bank address pins need to be connected starting with bit [0] and ending with the highest order bank address bit. Leave MSB unconnected if the DRAM has fewer bank address pins than the controller.

11.4.2 Programming DRAM Device Internal Configuration Register

- Set burst type to sequential
- Burst length is always 16-byte. Means 4-beat bursts in a 32-bit system, 8-beat burst in a 16-bit system
- Set CAS latency to lowest value DRAM can tolerate at intended speed, and then set write latency and read latency accordingly.
- Set posted CAS additive latency to 0
- Controller never uses auto-precharge on read or write.
- Configure DQS operation for single-ended operation.
- RTT and output drive strength configuration depends on electrical characteristics.

11.4.3 DRAM Command Engine

This block decides what command to send to the DRAM controller next. There are four different commands that can be sent to the DRAM to service incoming requests from the four incoming buses.

- Precharge
- Activate
- Read
- Write

On every rising clock edge, the DRAM command engine first determines using parallel logic what is highest priority pending precharge, activate, read and write command. Next, it decides which of these commands to send to the DRAM.

The arbiters that make the decisions about what command to send next to the DRAM are aware of the current state the DRAM is in. When arbitrating a command on the DRAM bus, the following information is processed:

- For each bank, if it is precharged or not
- For each incoming request, if it hits in an already active bank or not
- For each bank, if the DRAM currently can accept a precharge command to it
- For each bank, if the DRAM currently can accept an activate command to it
- For each bank, if the DRAM currently can accept a read command to it

- For each bank, if the DRAM currently can accept a write command to it

The logic keeping track of what is currently possible on each of the banks is not in the DRAM command engine. It is part of the timing manager, whose task is to signal to the DRAM command engine that commands are currently possible.

11.4.4 Write Buffer

All incoming writes are sent first to the write buffer, part of the command engine. Writes are sent to the DRAM in background, whenever possible. The DRAM tries to postpone the writes until there are no further outstanding read requests. However, when the write buffer is full, or when there is a new request for an address already inside the write buffer, the DRAM controller writes the content of the write buffer to the DRAM.

11.4.5 Timing Manager

The timing manager consists of a bank of counters. These counters keeps track of all DRAM timing parameters and signals to the DRAM command engine when a precharge, activate, read or write command is possible. This information is supplied to the DRAM command engine for each bank separately.

All timing parameters are programmable in software.

11.4.6 DRAM Read Block and DRAM Write Block

Sending a read or write command to the DRAM is a two-step process. First, the command is sent, which is done by the command engine. After some clock cycles, the data must follow.

Manipulating the read data is done by the read block. For every read command sent to the DRAM, the command engine informs the read block. Upon receiving the read command, the read block delays this to account for DRAM pipelining. Then, it receives the correct amount of data from the DRAM DQ inputs and forwards this data to the correct bus.

Manipulating the write data is done by the write block. It works the same way as the read block. The command engine informs the write block of a pending write. Upon receiving the command, the write block delays this to account for DRAM pipelining. Then, it receives the relevant data from the write buffer and transmits this to the DRAM.

11.4.7 Bus Interface

The bus interface accepts a slave peripheral bus. The bus interface fulfills several functions:

- It contains all configuration registers
- It contains logic to send an error interrupt to the processor. The error interrupt is active when the FIFO overflow or FIFO underflow error condition and corresponding interrupt enable in register DDR_SYS_CONFIG is set. The register summary is given in Table 11-1.

  The FIFO overflow and underflow flags are tied to a FIFO that keeps track of the number of DQS strobes the DRAM is expected to produce. If a read command is sent to the DRAM, the DRAM is expected to answer after producing the read data on its DQ outputs, with some edges on its DQS

output used by the controller to clock the read data. If the DRAM controller produces the read strobes at an incorrect time, or produces not enough or too many read strobes, the DRAM controller may detect some error conditions because they result in an overflow or underflow of the FIFO that keeps track of the number of outstanding DQS pulses. These bits do not detect timing configuration errors. Underflows and overflows signaled by the read FIFO point to following possible error sources:

  - Incorrect configuration of the DRAM. Burst length set incorrectly
  - Incorrect configuration of the DRAM controller.
    - Incorrect RDLY
    - Incorrect HALF_DQS_DLY
    - Incorrect QUART_DQS_DLY
    - Incorrect DRAM timing parameters or mis-match between various settings.
  - Problems with the electrical connections between the DRAM controller and the DRAM

- It contains a bypass path to send commands to the DRAM. This is because the DRAM controller contains no logic to take care of DRAM initialization, programming the mode registers, or putting the DRAM into or out of the sleep and standby modes like self-refresh. Essentially, these functions are made available over the peripheral bus. To program the mode registers, the DRAM controller needs to be put in a bypass mode, where incoming requests are not serviced. In this bypass mode, commands are sent from the peripheral interface directly to the DRAM to program the mode registers or to put the DRAM into or out of sleep mode.
- During bypass mode, all reads and writes are blocked. Refresh keeps running, but can be separately disabled.

This page is intentionally left blank.

Chapter 12
Multi-port DRAM Controller Priority Manager

12.1 Introduction

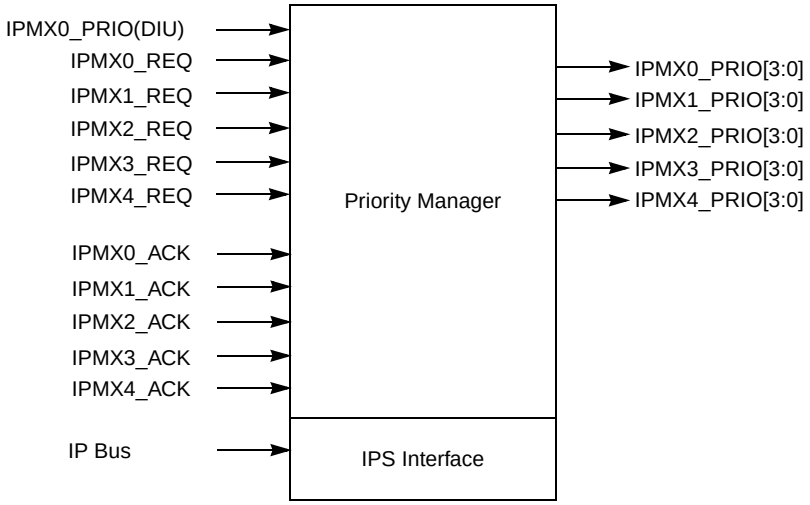


Figure 12-1. Priority Manager Block Diagram

The multi-port DRAM controller services the highest priority request from five different buses using a 4-bit priority signal. This 4-bit priority is dynamically set by the DRAM controller priority manager based on register settings and the most recent activity on each bus. In general, the DRAM priority manager increases the priority of a channel if it has not been recently serviced and decreases the priority of channels that have been recently serviced.

A block diagram of the priority manager is given in Figure 12-1. It accepts the request and ACK-signals for all five DRAM buses, and produces the priority signals for the five buses.

The priority manager uses an ACK-based schema; the priority depends on how many times for the last N requests accepted by the DRAM controller the current owner of the bus won the request. A running average counter keeps track of how many acknowledgements out of the last N acknowledgements have been for a specific bus. This number is then put in a look-up table configurable by writing some config registers. The output of the look-up table is the priority for the next request on this bus.

This priority schema is versatile because programming the look-up table allows controlling relative priority to other channels and the average share of the bandwidth the current master gets. The priority schema introduces fairness because the look-up table can be programmed to reduce the priority of a bus that has won a large share of requests and increase the priority of a bus that lost a large share of requests.

12.1.1 Features

- Dynamic priority calculation based on ACKing history
  - — Fully programmable using look-up table
    - – Can be configured for high or low latency and high or low bandwidth
    - – Separate control over average latency and average bandwidth
    - – Versatile so it can mimic the CSB arbitration schema
  - — Fairness guaranteed by reducing priority of channels that receive a lot of grants, and increasing priority of channels that are denied the bus often
  - — Repeat transfer built into the DRAM controller. Priority manager can set the maximum repeat count by controlling when lowest priority occurs.
- Feed-through mode where DIU priority is controlled directly by the DIU

12.2 Bus Connections

The following masters are connected to five buses.

- Bus 0: DIU
- Bus 1: Power architecture e300
- Bus 2: NFC
- Bus 3: Reserved
- Bus 4: USB, DMA, FEC

12.3 Memory Map and Register Definition

12.3.1 Memory Map

Table 12-1. PRIOMAN memory map

Offset from PRIOMAN_BASE (0xFF40_9000)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x0080	Priority Manager Configuration 1 Register (PRIOMAN_CONFIG1)	R/W	0x0007_7777	12.3.2.1/12-313
0x0084	Priority Manager Configuration 2 Register (PRIOMAN_CONFIG2)	R/W	0x0000_00U1	12.3.2.2/12-314
0x0088	High Priority Configuration Register (HIPRIO_CONFIG)	R/W	0x0000_UUUU	12.3.2.3/12-315
0x008C	Look-Up Table 0 Main Upper (LUT_TABLE0_MAIN_UP)	R/W	0x1111_1222	12.3.2.4/12-316
0x0090	Look-Up Table 1 Main Upper (LUT_TABLE1_MAIN_UP)	R/W	0x1111_1222	12.3.2.4/12-316
0x0094	Look-Up Table 2 Main Upper (LUT_TABLE2_MAIN_UP)	R/W	0x1111_1222	12.3.2.4/12-316
0x0098	Look-Up Table 3 Main Upper (LUT_TABLE3_MAIN_UP)	R/W	0x1111_1222	12.3.2.4/12-316
0x009C	Look-Up Table 4 Main Upper (LUT_TABLE4_MAIN_UP)	R/W	0x1111_1222	12.3.2.4/12-316

Table 12-1. PRIOMAN memory map (Continued)

Offset from PRIOMAN_BASE (0xFF40_9000)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x00A0	Look-Up Table 0 Main Lower (LUT_TABLE0_MAIN_LOW)	R/W	0x2334_567A	12.3.2.5/12-317
0x00A4	Look-Up Table 1 Main Lower (LUT_TABLE1_MAIN_LOW)	R/W	0x2334_567A	12.3.2.5/12-317
0x00A8	Look-Up Table 2 Main Lower (LUT_TABLE2_MAIN_LOW)	R/W	0x2334_567A	12.3.2.5/12-317
0x00AC	Look-Up Table 3 Main Lower (LUT_TABLE3_MAIN_LOW)	R/W	0x2334_567A	12.3.2.5/12-317
0x00B0	Look-Up Table 4 Main Lower (LUT_TABLE4_MAIN_LOW)	R/W	0x2334_567A	12.3.2.5/12-317
0x00B4	Look-Up Table 0 Alternate Upper (LUT_TABLE0_ALT_UP)	R/W	0x0000_0000	12.3.2.6/12-318
0x00B8	Look-Up Table 1 Alternate Upper (LUT_TABLE0_ALT_UP)	R/W	0x0000_0000	12.3.2.6/12-318
0x00BC	Look-Up Table 2 Alternate Upper (LUT_TABLE0_ALT_UP)	R/W	0x0000_0000	12.3.2.6/12-318
0x00C0	Look-Up Table 3 Alternate Upper (LUT_TABLE0_ALT_UP)	R/W	0x0000_0000	12.3.2.6/12-318
0x00C4	Look-Up Table 4 Alternate Upper (LUT_TABLE0_ALT_UP)	R/W	0x0000_0000	12.3.2.6/12-318
0x00C8	Look-Up Table 0 Alternate Lower (LUT_TABLE0_ALT_LOW)	R/W	0x0000_0000	12.3.2.7/12-319
0x00CC	Look-Up Table 1 Alternate Lower (LUT_TABLE0_ALT_LOW)	R/W	0x0000_0000	12.3.2.7/12-319
0x00D0	Look-Up Table 2 Alternate Lower (LUT_TABLE0_ALT_LOW)	R/W	0x0000_0000	12.3.2.7/12-319
0x00D4	Look-Up Table 3 Alternate Lower (LUT_TABLE0_ALT_LOW)	R/W	0x0000_0000	12.3.2.7/12-319
0x00D8	Look-Up Table 4 Alternate Lower (LUT_TABLE0_ALT_LOW)	R/W	0x0000_0000	12.3.2.7/12-319
0x00DC	Performance Monitor Configuration Register (PERF_MONITOR_CONFIG)	R/W	0x0800_0000	12.3.2.8/12-320
0x00E0	Event Time Counter (EVENT_TIME_COUNTER) Register	R/W	0x0000_0000	12.3.2.9/12-321
0x00E4	Event Time Preset (EVENT_TIME_PRESET) Register	R/W	0x0000_0000	12.3.2.10/12-322
0x00E8	Performance Monitor 1 Address Low (PERF_MNTR1_ADDR_LOW) Register	R/W	0x0000_0000	12.3.2.11/12-322
0x00EC	Performance Monitor 2 Address Low (PERF_MNTR2_ADDR_LOW) Register	R/W	0x0000_0000	12.3.2.11/12-322
0x00F0	Performance Monitor 1 Address High (PERF_MNTR1_ADDR_HI) Register	R/W	0x0000_0000	12.3.2.11/12-322
0x00F4	Performance Monitor 2 Address High (PERF_MNTR2_ADDR_HI) Register	R/W	0x0000_0000	12.3.2.11/12-322
0x0100	Performance Monitor 1 Read Counter (PERF_MNTR1_READ_CNTR)	R	0x0000_0000	12.3.2.12/12-323
0x0104	Performance Monitor 2 Read Counter (PERF_MNTR2_READ_CNTR)	R	0x0000_0000	12.3.2.12/12-323
0x0108	Performance Monitor 1 Write Counter (PERF_MNTR1_WRITE_CNTR)	R	0x0000_0000	12.3.2.12/12-323

Table 12-1. PRIOMAN memory map (Continued)

Offset from PRIOMAN_BASE (0xFF40_9000)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x010C	Performance Monitor 2 Write Counter (PERF_MNTR2_WRITE_CNTR)	R	0x0000_0000	12.3.2.12/12-323
0x0110	Granted Ack Counter 0 (GRANTED_ACK_CNTR0)	R	0x0000_0000	12.3.2.13/12-324
0x0114	Granted Ack Counter 1 (GRANTED_ACK_CNTR1)	R	0x0000_0000	12.3.2.13/12-324
0x0118	Granted Ack Counter 2(GRANTED_ACK_CNTR2)	R	0x0000_0000	12.3.2.13/12-324
0x011C	Granted Ack Counter 3 (GRANTED_ACK_CNTR3)	R	0x0000_0000	12.3.2.13/12-324
0x0120	Granted Ack Counter 4 (GRANTED_ACK_CNTR4)	R	0x0000_0000	12.3.2.13/12-324
0x0124	Cumulative Wait Counter 0 (CUMULATIVE_WAIT_CNTR0)	R	0x0000_0000	12.3.2.14/12-324
0x0128	Cumulative Wait Counter 1 (CUMULATIVE_WAIT_CNTR1)	R	0x0000_0000	12.3.2.14/12-324
0x012C	Cumulative Wait Counter 2 (CUMULATIVE_WAIT_CNTR2)	R	0x0000_0000	12.3.2.14/12-324
0x0130	Cumulative Wait Counter 3 (CUMULATIVE_WAIT_CNTR3)	R	0x0000_0000	12.3.2.14/12-324
0x0134	Cumulative Wait Counter 4 (CUMULATIVE_WAIT_CNTR4)	R	0x0000_0000	12.3.2.14/12-324
0x0138	Summed Priority Counter 0 (SUMMED_PRIORITY_CNTR0)	R	0x0000_0000	12.3.2.15/12-325
0x013C	Summed Priority Counter 1 (SUMMED_PRIORITY_CNTR1)	R	0x0000_0000	12.3.2.15/12-325
0x0140	Summed Priority Counter 2 (SUMMED_PRIORITY_CNTR2)	R	0x0000_0000	12.3.2.15/12-325
0x0144	Summed Priority Counter 3 (SUMMED_PRIORITY_CNTR3)	R	0x0000_0000	12.3.2.15/12-325
0x0148	Summed Priority Counter 4 (SUMMED_PRIORITY_CNTR4)	R	0x0000_0000	12.3.2.15/12-325
0x014C–0x03FF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

3 In this column, the symbol “U” indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

12.3.2 Register Descriptions

12.3.2.1 Priority Manager Configuration 1 Register (PRIOMAN_CONFIG1)

Address: Base + 0x0080 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LUT SEL4[1:0]		LUT SEL3[1:0]		LUT SEL2[1:0]		LUT SEL1[1:0]		LUT SEL0[1:0]		ACK_COUNT4[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ACK_COUNT3[2:0]				ACK_COUNT2[3:0]				ACK_COUNT1[3:0]				ACK_COUNT0[3:0]			
W																
Reset	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1

Figure 12-2. Priority Manager Configuration 1 Register (PRIOMAN_CONFIG1)

Table 12-2. PRIOMAN_CONFIG1 field descriptions

Field	Description
LUT SEL*n*	Lookup Table Select. Selects between primary and secondary Look-Up table configuration register. 00 Select main look-up table configuration register. 01 Select alternate look-up table configuration register. 10 Select alternate look-Up table configuration register if congested flag is set. See Section 12.3.2.3, “High Priority Configuration Register (HIPRIO_CONFIG).” 11 Select alternate look-up table configuration register if DIU incoming priority bit 3 is high. The switch for all tables is based on the DIU flag. If LUT_SEL*n* = 11, the e300 table switches to the alternate table if the DIU flag is set.
ACK_COUNT*n* [3:0]	Configuration fields. One for every channel. Determines how many requests the number of ACKs for the self-channel is counted.[1] 0000 1 ACK is counted. 0001 2 ACKs are counted. 0010 3 ACKs are counted. 0011 4 ACKs are counted. 0100 6 ACKs are counted. 0101 8 ACKs are counted. 0110 12 ACKs are counted. 0111 16 ACKs are counted. 1000 24 ACKs are counted. 1001 32 ACKs are counted. 1010 48 ACKs are counted. 1011 63 ACKs are counted.

[1] Look-up table input is running average of the number of acks for the self channel counted over the grant total of the last N acks. ACK_COUNT[2:0] controls the parameter N.

12.3.2.2 Priority Manager Configuration 2 Register (PRIOMAN_CONFIG2)

Address: Base + 00084 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	CONGESTED	DIU OVERFULL	ACKSEL4	ACKSEL3	ACKSEL2	ACKSEL1	ACKSEL0
W																
Reset	0	0	0	0	0	0	0	0	0	—	0	1	0	0	0	1

Figure 12-3. Priority Manager Configuration 2 Register (PRIOMAN_CONFIG2)

Table 12-3. PRIOMAN_CONFIG2 field descriptions

Field	Description
CONGESTED	Read only 0 Congested flag is cleared. 1 Congested flag is set.
DIU-OVER RULE	0 DIU priority follows normal schema. 1 Priority for channel 0 taken from DIU directly.
ACK SEL*n*	There is one of these bits for each priority manager channel. They determine what happens if the current channel is not requesting. 0 No special overrule. Regulates default priority to high value. 1 If current channel is not requesting, every ACK for other channel is treated like an ACK for the current channel. Regulates default priority to low value.

12.3.2.3 High Priority Configuration Register (HIPRIO_CONFIG)

The High Priority Configuration Register (HIPRIO_CONFIG) controls the high priority detection logic. The hiprio detection logic detects what percentage of the requests ACKed by the DRAM controller are ACKed with a priority larger than eight.

Address: Base + 0x0088 Access: User read/write

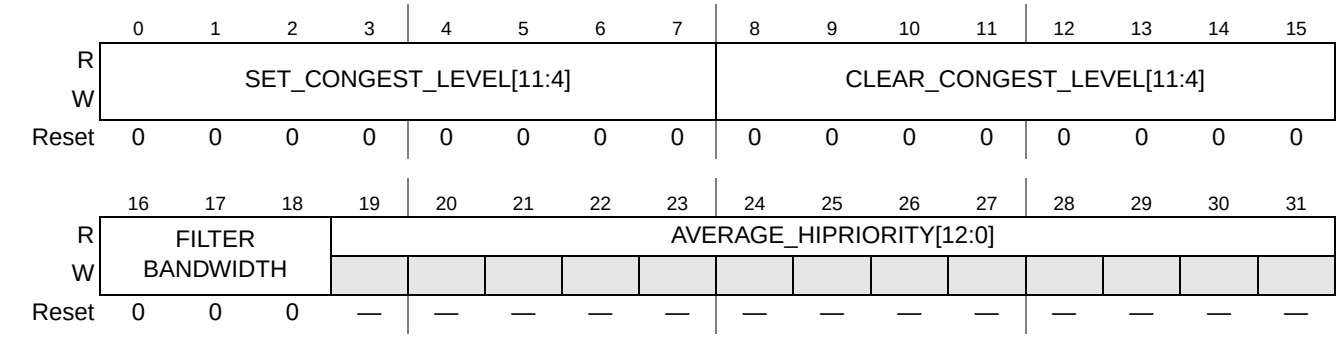


Figure 12-4. High Priority Configuration Register (HIPRIO_CONFIG)

Table 12-4. HIPRIO_CONFIG field descriptions

Field	Description
SET_CONGEST_LEVEL[11:4]	If(average_hipriority[12:4] > set_congest_level[12:4]) → set the congested flag.
CLEAR_CONGEST_LEVEL[11:4]	If(average_hipriority[12:4] < clear_congest_level[12:4]) → clear the congested flag.
AVERAGE_HIPRIORITY[12:0]	Average number of high priority requests to DRAM, coded between values 0x1000 and 0x0000 0x1000: 100% high-priority requests. 0x0000: 0% high-priority requests.
FILTER BANDWIDTH[2:0]	This setting controls the averaging time of the filter used for average_hipriority[12:0][1] 000 Time constant W0 = 8 ACKS, K = 0.125 001 Time constant W0 = 16 ACKS, K = 0.0625 010 Time constant W0 = 32 ACKS, K = 0.0312 011 Time constant W0 = 64 ACKS, K = 0.0156 100 Time constant W0 = 128 ACKS, K = 0.0078 101 Time constant W0 = 256 ACKS, K = 0.0039 110 Time constant W0 = 512 ACKS, K = 0.0020 111 Time constant W0 = 1024 ACKS, K = 0.0010

[1] Refer to Equation 12-1 and Equation 12-2 for the relationship between filter bandwidth and filter behavior.

12.3.2.4 Look-Up Table *n* Main Upper (LUT_TABLE*n*_MAIN_UP) Registers

The Look-Up Table *n* Main Upper (LUT_TABLE*n*_MAIN_UP) registers contain the upper eight entries of the look-up tables for channels 0 to 4, main table. All registers contain identical fields.

Address: Base + 0x008C (LUT_TABLE0_MAIN_UP)
Base + 0x0090 (LUT_TABLE1_MAIN_UP)
Base + 0x0094 (LUT_TABLE2_MAIN_UP)
Base + 0x0098 (LUT_TABLE3_MAIN_UP)
Base + 0x009C (LUT_TABLE4_MAIN_UP)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRIO15[3:0]				PRIO14[3:0]				PRIO13[3:0]				PRIO12[3:0]			
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRIO11[3:0]				PRIO10[3:0]				PRIO9[3:0]				PRIO8[3:0]			
Reset	0	0	0	1	0	0	1	0	0	0	1	0	0	0	1	0

Figure 12-5. Look-Up Table *n* Main Upper (LUT_TABLE*n*_MAIN_UP) Registers

Table 12-5. LUT_TABLE*n*_MAIN_UP field descriptions

Field	Description
PRIO15[3:0]	Priority setting if 15 or more ACKs for own channel counted
PRIO14[3:0]	Priority setting if 14 ACKs for own channel counted
PRIO13[3:0]	Priority setting if 13 ACKs for own channel counted
PRIO12[3:0]	Priority setting if 12 ACKs for own channel counted
PRIO11[3:0]	Priority setting if 11 ACKs for own channel counted
PRIO10[3:0]	Priority setting if 10 ACKs for own channel counted
PRIO9[3:0]	Priority setting if 9 ACKs for own channel counted
PRIO8[3:0]	Priority setting if 8 ACKs for own channel counted

12.3.2.5 Look-Up Table *n* Main Lower (LUT_TABLE*n*_MAIN_LOW) Registers

The Look-Up Table *n* Main Lower (LUT_TABLE*n*_MAIN_LOW) registers contain the lower eight entries of the look-up tables for channels 0 to 4, main table. All registers contain identical fields.

Address: Base + 0x00A0 (LUT_TABLE0_MAIN_LOW)
Base + 0x00A4 (LUT_TABLE1_MAIN_LOW)
Base + 0x00A8 (LUT_TABLE2_MAIN_LOW)
Base + 0x00AC (LUT_TABLE3_MAIN_LOW)
Base + 0x00B0 (LUT_TABLE4_MAIN_LOW)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRIO7[3:0]				PRIO6[3:0]				PRIO5[3:0]				PRIO4[3:0]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRIO3[3:0]				PRIO2[3:0]				PRIO1[3:0]				PRIO0[3:0]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-6. Look-Up Table *n* Main Lower (LUT_TABLE*n*_MAIN_LOW) Registers

Table 12-6. LUT_TABLE*n*_MAIN_LOW field descriptions

Field	Description
PRIO7[3:0]	Priority setting if seven ACKs for own channel counted
PRIO6[3:0]	Priority setting if six ACKs for own channel counted
PRIO5[3:0]	Priority setting if five ACKs for own channel counted
PRIO4[3:0]	Priority setting if four ACKs for own channel counted
PRIO3[3:0]	Priority setting if three ACKs for own channel counted
PRIO2[3:0]	Priority setting if two ACKs for own channel counted
PRIO1[3:0]	Priority setting if one ACK for own channel counted
PRIO0[3:0]	Priority setting if zero ACKs for own channel counted

12.3.2.6 LUT0 – LUT4 Alternate Upper

These registers contain the upper eight entries of the look-up tables for channels 0 to 4, alternate table. All registers contain identical fields.

Address: Base + 0x00B4 (LUT0 alternate upper)
Base + 0x00B8 (LUT1 alternate upper)
Base + 0x00BC (LUT2 alternate upper)
Base + 0x00C0 (LUT3 alternate upper)
Base + 0x00C4 (LUT4 alternate upper)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRIO15[3:0]				PRIO14[3:0]				PRIO13[3:0]				PRIO12[3:0]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRIO11[3:0]				PRIO10[3:0]				PRIO9[3:0]				PRIO8[3:0]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-7. LUT – LUT4 Alternate Upper Register

Table 12-7. LUT Table Alternate Upper field descriptions

Field	Description
PRIO15[3:0]	Priority setting if 15 or more ACKs for own channel counted
PRIO14[3:0]	Priority setting if 14 ACKs for own channel counted
PRIO13[3:0]	Priority setting if 13 ACKs for own channel counted
PRIO12[3:0]	Priority setting if 12 ACKs for own channel counted
PRIO11[3:0]	Priority setting if 11 ACKs for own channel counted
PRIO10[3:0]	Priority setting if 10 ACKs for own channel counted
PRIO9[3:0]	Priority setting if 9 ACKs for own channel counted
PRIO8[3:0]	Priority setting if 8 ACKs for own channel counted

12.3.2.7 LUT0 – LUT4 Alternate Lower

These registers contain the lower eight entries of the look-up tables for channels 0 to 4, alternate table. All registers contain identical fields.

Address: Base + 0xHHHH — Access: User read/write

0xC8	LUT0 alternate lower
0xCC	LUT1 alternate lower
0xD0	LUT2 alternate lower
0xD4	LUT3 alternate lower
0xD8	LUT4 alternate lower

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRIO7[3:0]				PRIO6[3:0]				PRIO5[3:0]				PRIO4[3:0]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRIO3[3:0]				PRIO2[3:0]				PRIO1[3:0]				PRIO0[3:0]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-8. LUT Table [4:0] Alternate Lower Register

Table 12-8. LUT Table Alternate Lower field descriptions

Field	Description
PRIO7[3:0]	Priority setting if seven ACKs for own channel counted
PRIO6[3:0]	Priority setting if six ACKs for own channel counted
PRIO5[3:0]	Priority setting if five ACKs for own channel counted
PRIO4[3:0]	Priority setting if four ACKs for own channel counted
PRIO3[3:0]	Priority setting if three ACKs for own channel counted
PRIO2[3:0]	Priority setting if two ACKs for own channel counted
PRIO1[3:0]	Priority setting if one ACK for own channel counted
PRIO0[3:0]	Priority setting if zero ACKs for own channel counted

12.3.2.8 Performance Monitor Configuration (PERF_MONITOR_CONFIG) Register

Address: Base + 0x00DC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT		INTEN	DMA REQ	DMA REQ STOP	EVENTCO UNTFREE RUN		0	0	0	0	0	0	0	0	0
W		INTCLEAR					EVENTCO UNTTRIG									
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	LUT SEL4		LUT SEL3		LUT SEL2		LUT SEL1		LUT SEL0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-9. Performance Monitor Configuration (PERF_MONITOR_CONFIG) Register

Table 12-9. PERF_MONITOR_CONFIG field descriptions

Field	Description
INT	Read-Only Sticky Bit. Interrupt pending register. Set when interrupt is made pending.
INTCLEAR	Write-Only. Writing a 1 to this bit clears the INT bit.
INTEN	Interrupt Enable. When this bit is 1 and the INT bit is 1, the processor gets an interrupt request.
DMAREQ	DMA request. This read-only bit is set when event counter time reaches 0, and cleared when the first counter register is read.
DMAREQSTOP[1]	Read/Write. 0 DMA request functions as expected. 1 DMA request is cleared and cannot get set.
EVENTCOUNT FREERUN	0 Event Counter Single-Shot. After reaching 0, the event time counter stays at 0 and is not reloaded. 1 Event Counter Free Run. After reaching 0, the event time counter is reloaded from event time preset and a new cycle starts.
EVENTCOUNT TRIGGER	Write-Only Bit. Writing to this bit causes all count registers to be transferred to the buffer registers, and subsequent be cleared. It causes the event counter to be reloaded from the event time preset register. No interrupt or DMA request is generated on writing this register, but both are generated when the event time counter register reaches 0.
LUT SEL*n*	Selectors between primary and secondary look-up table configuration register. These selectors determine which LUT table is used for the summed priority counters. The priorities entered into these counters may depend on a different LUT table than the priorities sent to the DRAM controller. 00 Select main look-up table configuration register 01 Select alternate look-up table configuration register 10 Select alternate look-up table configuration register if congested flag is set[2] 11 Select alternate look-up table configuration register if DIU incoming priority bit 3 is high

[1] This bit should be set as long as the DMA channel is not configured to manage the request. After configuring the DMA, clear the bit, and data starts to be transferred on every time tick.

[2] Congested flag is explained in Section 12.4.3, “Congestion Detector.”

12.3.2.9 Event Time Counter (EVENT_TIME_COUNTER) Register

The Event Time Counter (EVENT_TIME_COUNTER) register contains the 24-bit EVENT_TIME_COUNTER bitfield. The counter decrements to 0. The interrupt and the DMA request are made pending when it reachers 0. On reaching 0, the counter reloads from the event count preset register if the EVENTCOUNTFREERUN bit is set in the PERMON_CONFIG register (see Section 12.3.2.8, “Performance Monitor Configuration (PERF_MONITOR_CONFIG) Register).

On reaching 0, all performance monitor count registers are loaded in the performance monitor buffer registers, and cleared.

Address: Base + 0x00E0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	EVENT_TIME_COUNTER[23:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EVENT_TIME_COUNTER[23:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-10. Event Time Counter (EVENT_TIME_COUNTER) Register

Table 12-10. EVENT_TIME_COUNTER field descriptions

Field	Description
EVENT_TIME_COUNTER[23:0]	Current count in the register.

12.3.2.10 Event Time Preset (EVENT_TIME_PRESET) Register

The Event Time Preset (EVENT_TIME_PRESET) register contains the 24-bit preset value to be loaded into the event time counter register in case this preloads.

Address: Base + 0x00E4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	EVENT_TIME_PRESET[23:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EVENT_TIME_PRESET[23:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-11. Event Time Preset (EVENT_TIME_PRESET) Register

Table 12-11. EVENT_TIME_PRESET field descriptions

Field	Description
EVENT_TIME_PRESET[23:0]	Current preset value in the register.

12.3.2.11 Performance Monitor 1 and 2 Address Registers

These registers determine if a Power Architecture (e300) processor access hits in the performance monitor 1 or performance monitor 2 address space.

- If ((e300 CPU address ≥ performance monitor 1 address low) &&
  (e300 CPU address < performance monitor 1 address hi))
  Increment *performance monitor 1 read counter* on reads
  Increment *performance monitor 1 write counter* on writes.
- If ((e300 CPU address ≥ performance monitor 2 address low) &&
  (e300 CPU address < performance monitor 2 address hi))
  Increment *performance monitor 2 read counter* on reads
  Increment *performance monitor 2 write counter* on writes.

Address: Base + 0x0E8 (PERF_MNTR1_ADDR_LOW)
Base + 0x0EC (PERF_MNTR2_ADDR_LOW)
Base + 0x0F0 (PERF_MNTR1_ADDR_HI)
Base + 0x0F4 (PERF_MNTR2_ADDR_HI)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PERFORMANCE MONITOR {1,2} ADDRESS {LOW,HIGH}[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PERFORMANCE MONITOR {1,2} ADDRESS {LOW, HIGH} [15:5]											0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-12. Performance Monitor Address Registers

Table 12-12. Performance Monitor Address Registers field descriptions

Field	Description
PERFORMANCE MONITOR ADDRESS	

12.3.2.12 Performance Monitor Counters

Address: Base + 0x100 (Performance monitor 1 read counter)
Base + 0x104 (Performance monitor 2 read counter)
Base + 0x108 (Performance monitor 1 write counter)
Base + 0x10C (Performance monitor 2 write counter)

Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PERFORMANCE MONITOR 1, 2 READ/WRITE COUNTER[23:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PERFORMANCE MONITOR 1, 2 READ/WRITE COUNTER[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-13. Performance Monitor Counter Registers

Table 12-13. Performance Monitor Counter Register field descriptions

Field	Description
PERFORMANCE MONITOR READ/WRITE COUNTER[23:0]	

12.3.2.13 Granted Ack Counters

Address: Base + 0x110 (GRANTED_ACK_CNTR0)
Base + 0x114 (GRANTED_ACK_CNTR1)
Base + 0x118 (GRANTED_ACK_CNTR2)
Base + 0x11C (GRANTED_ACK_CNTR3)
Base + 0x120 (GRANTED_ACK_CNTR4)

Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	GRANTED ACK COUNTER 0 – 4[23:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GRANTED ACK COUNTER 0 – 4[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-14. Granted ACK Counter 0–4 Registers

Table 12-14. GRANTED_ACK_CNTR*n* field descriptions

Field	Description
GRANTED ACK COUNTER*n*[23:0]	

12.3.2.14 Cumulative Wait Counters

Address: Base + 0x124 (CUMULATIVE_WAIT_CNTR0)
Base + 0x128 (CUMULATIVE_WAIT_CNTR1)
Base + 0x12C (CUMULATIVE_WAIT_CNTR2)
Base + 0x130 (CUMULATIVE_WAIT_CNTR3)
Base + 0x134 (CUMULATIVE_WAIT_CNTR4)

Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	CUMULATIVE WAIT COUNTER 0 – 4[23:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CUMULATIVE WAIT COUNTER 0 – 4[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-15. Cumulative Wait Counter 0–4 Registers

Table 12-15. CUMULATIVE_WAIT_CNTR*n* Registers field descriptions

Field	Description
CUMULATIVE WAIT COUNTER*n*[23:0]	

12.3.2.15 Summed Priority Counters

Address: Base + 0x138 (SUMMED_PRIORITY_CNTR0)
Base + 0x13C (SUMMED_PRIORITY_CNTR1)
Base + 0x140 (SUMMED_PRIORITY_CNTR2)
Base + 0x144 (SUMMED_PRIORITY_CNTR3)
Base + 0x148 (SUMMED_PRIORITY_CNTR4)

Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SUMMED PRIORITY COUNTER 0 – 4[23:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SUMMED PRIORITY COUNTER 0 – 4[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-16. Summed Priority Counter 0–4 Registers

Table 12-16. SUMMED_PRIORITY_CNTR*n* Registers field descriptions

Field	Description
SUMMED PRIORITY COUNTER*n*[23:0]	

12.3.2.16 Counter Register Descriptions and Values

The counter registers contain 19 different 24-bit counter values. All these counter values count certain events. Table 12-17 gives the details on the nature of the event. Counter values Summed Priority Counter 2, Summed Priority Counter 3 and Summed Priority Counter 4 are available in two sets of registers. They are available in registers with the same name, but they are also available in a set of three other registers (granted ack counter or cumulative wait counter registers). The multiple-mapping of the three upper Summed Priority Counter registers allows easy and compact DMA transfer to memory. Because of the multiple mapping, all 19 count values can be transferred to memory with a 64-byte DMA transfer starting at address 0x100. The multiple mapping allows the DMA to get all information with a 64-byte transfer, but some decompression is needed on decoding the data, while the CPU can read the 19 registers and mask out the upper eight bits to get relevant information.

Table 12-17. Monitor Counter Descriptions

Field	Description
PERFORMANCE MONITOR 1–2 READ COUNTER	Every time the Processor performs a read access with an address that hits in the address window for counter 1 or 2, the respective counter is incremented. An address hits in the address window for performance monitor read counter 1 if the address is higher or equal than the performance monitor 1 address low, and lower than the performance monitor 1 address high. Similar for the second counter.
PERFORMANCE MONITOR 1–2 WRITE COUNTER	Every time the Processor performs a write access with an address that hits in the address window for counter 1 or 2, the respective counter is incremented. An address hits in the address window for performance monitor write counter 1 if the address is higher or equal than the performance monitor 1 address low, and lower than the performance monitor 1 address high. Similar for the second counter.
GRANTED ACK COUNTER 0–4	Every time the Multi-port DRAM controller grants a request for channel 0 – 4, the respective counter is incremented.
CUMULATIVE WAIT COUNTER 0–4	Every time there is a request pending to the multi-port DRAM controller for channel 0 – 4 and its not granted in the current cycle, the respective counter is incremented.
SUMMED PRIORITY COUNTER 0–4	Every time a request is granted by the multi-port DRAM controller for channel 0 – 4, a priority code is added to the respective counter. See text for details.

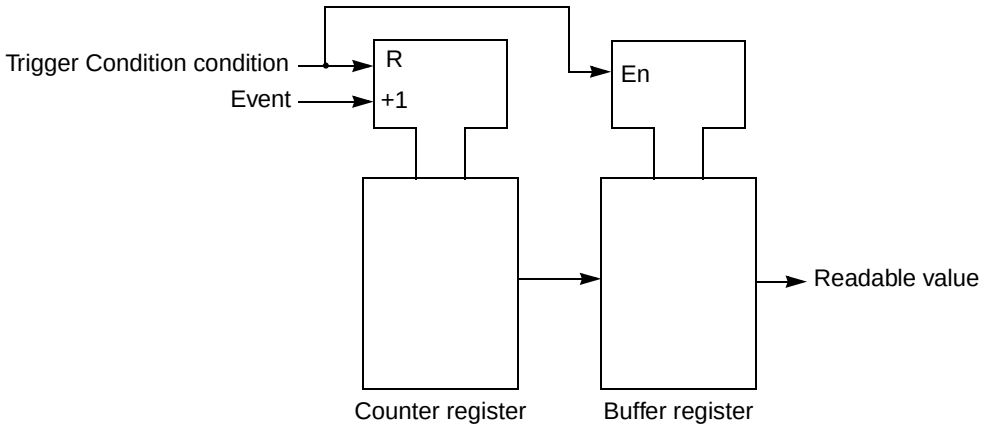


Figure 12-17. Monitor Counters

All counters in Table 12-17 are double-buffered and Figure 12-17 gives details. There are always two registers associated with every counter. The first register is the counter. It counts the events mentioned in the table. When the trigger condition occurs, the time event counter reaches zero, and the counter register is transferred to the buffer register, the counter register is then cleared. When accessing the register, the buffer register value is always returned.

The priority code added may or may not be the same as the priority code the request used on the DRAM controller. The codes are equal if the LUT SEL bitfield for the channel is the same in the PRIOMAN_CONFIG and PERMON_CONFIG registers. If the LUT SEL bitfields differ, the bitfield in PRIOMAN_CONFIG is used to calculate the channel priority code on the DRAM, and the bitfield in PERMON_CONFIG is used to calculate the priority code added to this register.

The possibility to use unequal LUT SEL bitfields makes it possible to use the main look-up tables for DRAM priority programming and the alternate look-up tables for performance monitoring. Making the look-up tables independent increases the possibility of what can be monitored.

12.4 Functional Description

The priority manager calculates the outgoing priority for all five channels of multi-port DRAM controller. The priority of any channel at a given time is a function of the request granting history of the DRAM controller. A granted request is called an ACK, so this schema is called an ACK-based schema, because the priority is determined by the history of which channels have been ACK-ed in the past and when.

The priority manager calculates the priorities in a dynamic way. This means, a priority is never constant, but changes over time, even when the request is not serviced. As a request ages while its not being serviced, its priority escalates to a higher level, and as the level increases, it is eventually serviced.

The DRAM controller has a built-in preference to offer repeat for any incoming read request. The repeat goes on as long as the requesting channel keeps requesting, and its priority is greater than 0. When the outgoing priority for any channel is 0, the DRAM controller no longer services or repeats the request. This feature allows the priority manager to control the maximum repeat count for any incoming channel.

12.4.1 Description of Operation — Overview

Priority calculation for all channels is independent. There is no direct cross-dependency of the priority of one channel on the priority of another channel. The algorithm looks at the last N arbitration cycles on the bus. N is a programmable number, set by the ACK_COUNT*n* bitfields in the PRIOMAN_CONFIG1 register, described in Figure 12-2. For the last N arbitration cycles, the number of times the own channel won the bus, is summed up, and saturated to a maximum of 15. This number of 0 to 15 is input into the applicable look-up table. LUT table 0 is for channel 0, LUT table 1 is for channel 1, and so on. The value for the particular number is the priority code going to the multiport DRAM controller. If N is set to 16 and the own channel was granted the bus four times in the last 16 bus grant, the index into the look-up table is four. The PRIO4[3:0] bitfield of the relevant look-up table is the priority going to the multi-port DRAM controller.

There are two look-up tables for every channel, the main and the alternate. The algorithm may switch between both, depending on some settings. The default look-up table is the main. However, the alternate is used if:

- The particular channel has been configured to look at the DIU incoming priority, and the DIU incoming priority is eight or higher.
- The particular channel has been configured to look at the congestion monitor, and this block indicates the multi-port DRAM is congested.

12.4.2 Block Diagram

Figure 12-18 contains a block diagram of the block.

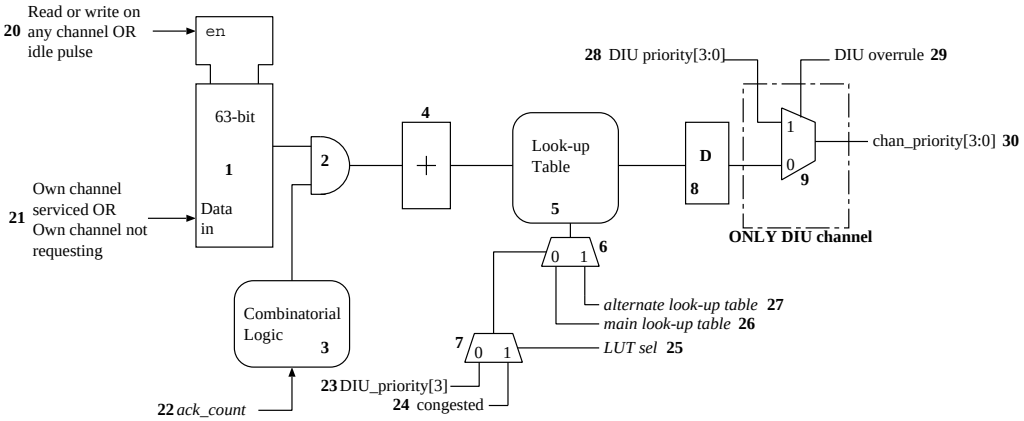


Figure 12-18. Priority Channel Block Diagram

The shift register shifts in information of the recent ACKs. This 63-stage shift register contains information on the last 63 bus cycles of the DRAM controller.

The shift register is shifted any time a read or write request has been granted to the DRAM (An ACK to the requesting bus) or when there is an IDLE_PULSE. An idle pulse is generated every time the DRAM is idle for four consecutive clock cycles. Idle means none of the five incoming buses is making a request.

The shift data in is the corrected ACK for the self channel. If the shift register shifts because the current cycle is granted to the self channel, a 1 is shifted in, if not a 0 is shifted in. It always occurs like this when the own channel is requesting access. However, if the own channel is not requesting access, depending on control bit ACK_SEL, a 1 or a 0 is shifted in. If ACK_SEL is 1, a 1 is shifted in all the time when the self channel is not requesting and there is an ACK on any other channel or an idle pulse. If ACK_SEL is 0, zeros are shifted in.

The correction for the non-requesting channel allows you to steer the default priority, the priority that the channel gets, when it has not been requesting for some time. If ACK_SEL is set 1, the default priority is low. This setting is appropriate for peripherals with (large) FIFOs. When they are not requesting, the FIFO is quite full. When they do get on the bus, they can start with low priority and escalate to higher after some time.

Setting ACK_SEL to 0 is appropriate for peripherals that desire high priority. In this case, the Power Architecture processor should receive high priority. When it is not on the bus, it is because it finds the instruction or data that it needs in the processor caches, so it does not request. When the cache misses, the request comes on the bus, and needs to be serviced fast. Therefore, ACK_SEL is set to 0, the default priority is high and servicing fast. If the Power Architecture Processor gets on the bus a lot (due to a lot of cache swapping), the priority manager detects this and degrades its priorities over time. The other masters continue to receive their fair share of bus bandwidth.

The output of the shift register is ANDED in to look at only the last *n* ACKs. Logic decodes the ANDing code from the ACK_COUNT bitfield. The number of ones after the ANDing is added up in ADDER 4 and saturated. The result out of ADDER 4 is a number from 0 to 15. This number is input in the look-up table. Table look-up content is taken for channel 1 from register lut table 1 main[63:0] or lut table 1

alternate[63:0]. Because of the 64-bit nature of the registers, four 32-bit registers are involved. The description is given in Figure 12-5, Figure 12-6, Figure 12-7, and Figure 12-8.

The MUX selects whether to use the main or the alternate register. The MUX condition has two possible sources again, selected by MUX **7**, by means of control bit LUT SEL described in the PRIOMAN_CONFIG register, with details in Figure 12-2.

If LUT SEL is 1, the alternate table is selected when the multi-port controller is congested. If LUT SEL is 0, the alternate table is selected when DIU incoming priority is higher than eight.

Pipeline register is present purely for implementation reasons. It has no algorithmic function.

For the DIU, an additional bypass multiplexer is present. It overrules the prioman logic and inserts the incoming DIU priority in the output if control bit DIU overrule is set. This bit is present in register PRIOMAN_CONFIG, with details in Figure 12-2.

12.4.3 Congestion Detector

The congestion detectors purpose is to detect when the multi-port DRAM controller is congested. Congestion is assumed if the share of the requests with priorities equal or greater than eight is more than a certain percentage. If congestion occurs, the priority manager may react by exchanging the look-up tables with the alternate look-up tables. This reduces the average priority of the incoming requests. The reduced priorities mean that on average, every incoming channel gets a lower priority and the DRAM controller tries harder to optimize on bandwidth and less to optimize to service the high-priority requests first. The switch-over is driven by the congestion state. If many requests come in on high priority, they all need to be serviced first, the congested flag goes high, and the controller reacts to this by reducing the request priorities (by switching in the alternate tables). Therefore, it can concentrate on the ones that are important and have room again for optimized bandwidth.

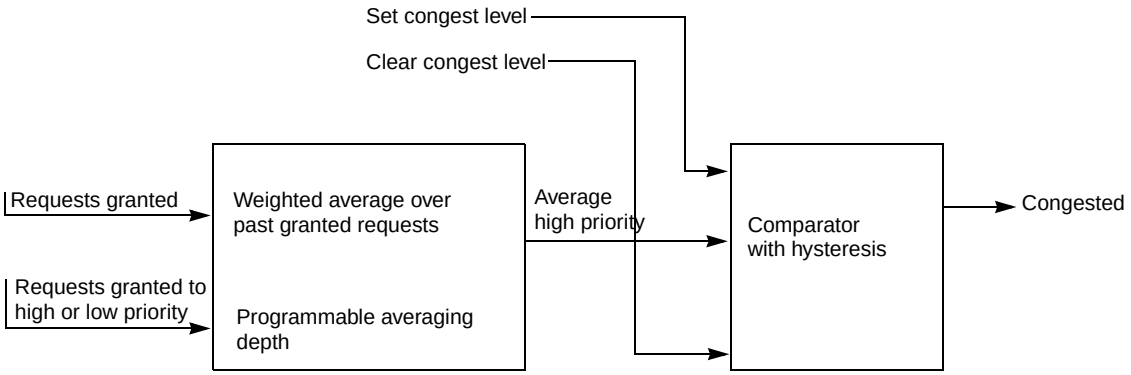


Figure 12-19. Congestion Detector—Simplified Block Diagram

A block diagram of the congestion detector is given in Figure 12-19. The congestion detector consists of an averaging block, followed by a comparator with hysteresis. The averaging block calculates the weighted average of the percentage of high-priority requests, like given below.

$$\text{average priority} = \sum \text{weight}(k) \cdot \text{val}(k) \qquad \textit{Eqn. 12-1}$$

The weighted average uses an exponential weighting when looking at the past granted requests. Requests granted in a more distant past have a lower weighting coefficient. The weighting coefficient uses an exponential back-off, following the formulae.

$$\text{weight}(k) = \frac{1}{W0} \cdot \exp\left(-\frac{k}{W0}\right) \qquad \textit{Eqn. 12-2}$$

In this formula, *weight(k)* is the weighting coefficient used for the request granted k acknowledges ago, meaning k other requests have been granted after this one. The coefficient W0 is programmable and depends on the control field filter bandwidth in register HIPRIO_CONFIG, detailed in Figure 12-4.

The value input in the weighting block, *val(k)*, depends on the priority of the request granted. It is 0 if the priority was 7 or lower; it is 0x1000 if the priority was 8 or higher.

The result of the weighted average is a number between 0 and 0x1000 input in the comparator with hysteresis. This result, AVERAGE_HIPRIORITY, can be monitored in register HIPRIO_CONFIG (Figure 12-4).

The weighted averaging block is followed by a comparator with hysteresis, with a programmable low threshold.

- If AVERAGE_HIPRIORITY is greater than SET_CONGEST_LEVEL, the congested flag is set.
- If AVERAGE_HIPRIORITY is lower than CLEAR_CONGEST_LEVEL, the congested flag is cleared.

Chapter 13
External Memory Bus (EMB)

13.1 Introduction

13.1.1 Overview

The external memory bus (EMB) includes two different parallel interfaces, the LocalPlus bus and the NAND flash bus. The two buses are time multiplexed.

An EMB arbiter controls the multiplexing of the external pins (address and data lines) and grants the different bus masters to allow them to drive the external bus. The arbiter can be configured via the EMB Share and Wait Count (LPC_EMB_SC) Register and EMB Pause Control (LPC_EMB_PC) Register within the LPC memory map.

13.1.2 Features

- Arbitration between LPC and NFC
  - LPC CSB transfers cannot be paused.
  - LPC CSB request pauses NFC transaction immediately or after share counter expires.
  - LPC FIFO request pauses NFC transaction after share counter expires
- Pin muxing between LPC and NFC

13.2 Functional Description

13.2.1 EMB Mux

The EMB mux switches, depending on EMB arbiter state, between the two different functions. The activated, granted module can drive the external bus.

Table 13-2 describes which functionality is at the EMB bus depending on the activated, granted module.

Table 13-2. EMB_AD Multiplexing

Activated, Granted Module	Multiplexed Functionality at EMB_AD[31:16]	Multiplexed Functionality at EMB_AD[15:0]	Multiplexed Functionality at EMB_AX[2:0]
LPC	LPC_AD[31:16]	LPC_AD[15:0]	LPC_AX[2:0]
NFC	12'H000,NFC_ALE,NFC_CLE, NFC_RE,NFC_WE	NFC_AD[15:0]	

This page is intentionally left blank.

Chapter 14
Fast Ethernet Controller (FEC)

14.1 Introduction

14.1.1 FEC Top Level

The block diagram of the FEC is shown in Figure 14-1. To implement the FEC, a combination of hardware and microcode is employed. The network interfaces are shown on the bottom of the diagram, complying with industry and IEEE 802.3 standards.

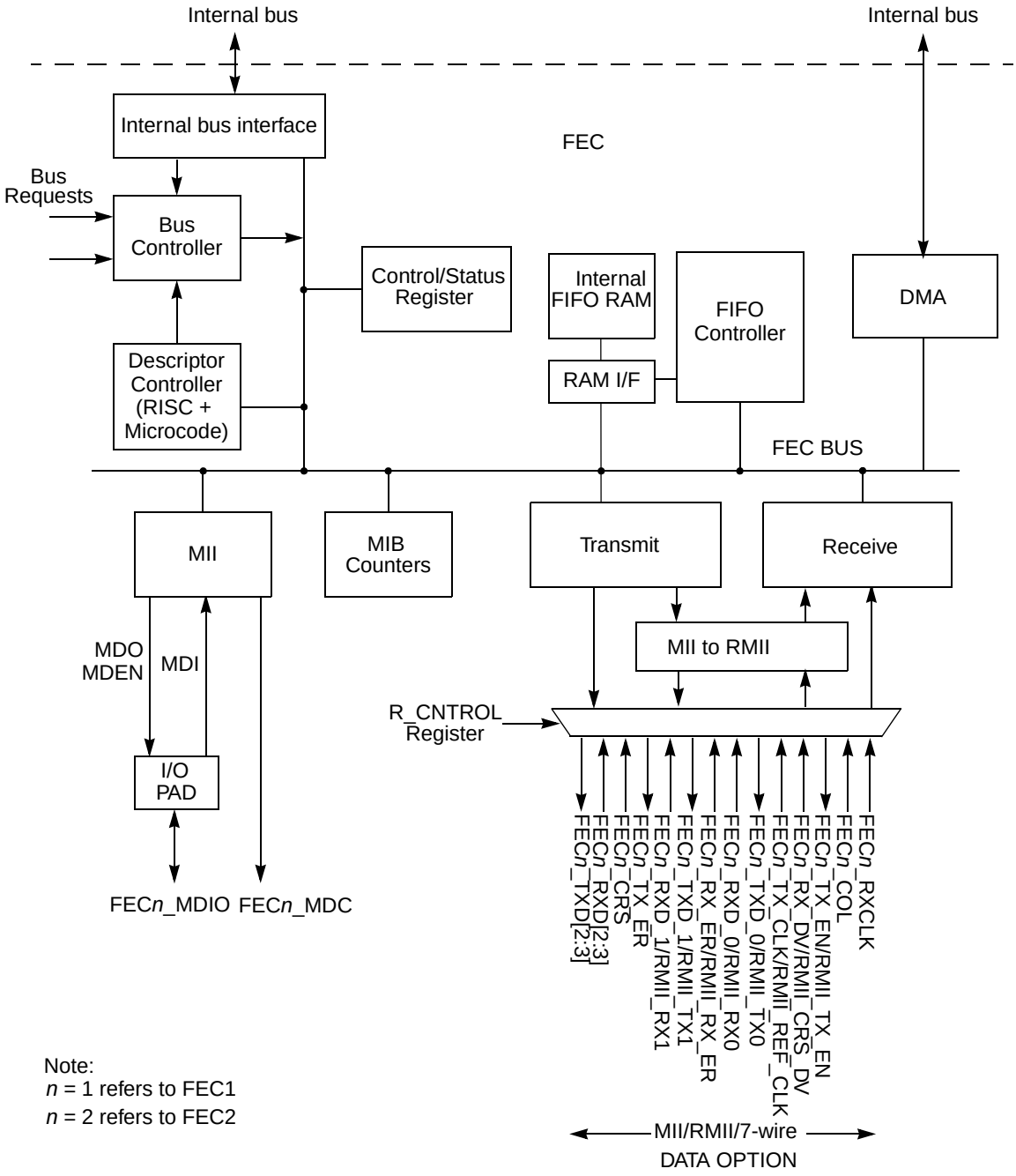


Figure 14-1. FEC Block Diagram

A RISC-based controller, called the descriptor controller, provides the following functions in the FEC:

- Initialization (those internal registers not initialized by the user or hardware)
- High-level control of the DMA channels (initiating DMA transfers)
- Interpreting buffer descriptors
- Address recognition for receive frames
- Random number generation for transmit collision backfill timer

NOTE

DMA references in this section refer to the FEC's DMA engine. This DMA engine transfers FEC data only and is not related to the DMA controller in MPC5125.

NOTE

The FIFO is used by FEC itself and can only be accessed by the DMA. You can configure the transmit/receive FIFO boundary (ETH_R_FSTART register).

The RAM is the central point of all data flow in the Fast Ethernet controller. The RAM is divided into transmit and receive FIFOs and the boundary is programmable (ETH_R_FSTART register). User data flows to/from the DMA unit from/to the receive/transmit FIFOs. Transmit data flows from the transmit FIFO into the transmit block and receive data flows from the receive block into the receive FIFO.

The bus controller decides which block is the tbus master on each clock. All of the blocks receive their control information from the tbus and, for the most part, provide status information over this same bus.

The user controls FEC by writing into control registers located in each block. The CSR (control and status register) block provides global control (e.g., Ethernet reset and enable, mode control) and interrupt managing registers.

The MII block provides a serial channel for control/status communication with the external physical layer device (transceiver). This serial channel consists of the MDC (clock) and MDIO (bidirectional data) lines of the MII interface.

The FEC DMA block (not to be confused with DMA controller) provides multiple channels allowing transmit data, transmit descriptor, receive data and receive descriptor accesses to run independently.

The transmit and receive blocks provide the Ethernet MAC functionality (with some assist from microcode). Internal to these blocks are clock domain boundaries between the system clock and the network clocks.

The MIB block maintains counters for a variety of network events and statistics. It is not necessary for operation of the FEC but provides valuable counters for network management. The counters supported are the RMON (RFC 1757) Ethernet Statistics group and some of the IEEE 802.3 counters.

14.1.2 Features

The fast Ethernet controller (FEC) incorporates several features/design goals important to its market:

- Support for different Ethernet physical interfaces:
  - 100 Mbit/s IEEE 802.3 MII
  - 10 Mbit/s IEEE 802.3 MII
  - 100 Mbit/s reduced media independent interface (RMII)
  - 10 Mbit/s reduced media independent interface (RMII)
  - 10 Mbit/s 7-wire interface (industry standard)
- IEEE 802.3 full-duplex flow control

- Programmable maximum frame length supports IEEE 802.1 VLAN tags and priority
- Support for full-duplex operation (200 Mbit/s throughput) with a minimum system clock rate of 50 MHz
- Support for half-duplex operation (100 Mbit/s throughput) with a minimum system clock rate of 25 MHz
- Retransmission from transmit FIFO following a collision (no processor bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization)
  - Address recognition
  - Frames with broadcast address may always be accepted or always be rejected
  - Exact match for single 48-bit individual (unicast) address
  - Hash (64-bit hash) check of individual (unicast) addresses
  - Hash (64-bit hash) check of group (multicast) addresses
  - Promiscuous mode

14.1.3 Modes of Operation

The primary operational modes are described in this section.

- Full- and half-duplex operation

  This is determined by the FDEN bit in the ETH_X_CNTRL register. Full-duplex mode is intended for use on point-to-point links between switches or end node to switch. Half-duplex mode is used in connections between an end node and a repeater or between repeaters.

  Full-duplex flow control is an option that may be enabled in full-duplex mode. Refer to the RFC_PAUSE and TFC_PAUSE bits in Section 14.3.5.12, “Transmit Control (ETH_X_CNTRL) Register,” the FCE bit in Section 14.3.5.10, “Receive Control (ETH_R_CNTRL) Register,” and Section 14.6.4, “Full-Duplex Flow Control,” for more details.

- 10 Mbit/s and 100 Mbit/s MII interface operation

  The MAC-PHY interface operates in MII mode by asserting the MII_MODE bit in the ETH_R_CNTRL register. The MII is the media-independent interface defined by the 802.3 standard for 10/100 Mbit/s operation.

  The speed of operation is determined by the TX_CLK and RX_CLK pins, which are driven by the transceiver. The transceiver auto-negotiates the speed or may be controlled by software via the serial management interface (MDC/MDIO pins) to the transceiver. Refer to the ETH_MII_DATA and ETH_MII_SPEED register descriptions as well as the section on the MII for a description of how to read and write registers in the transceiver via this interface.

- 10 Mbit/s and 100 Mbit/s RMII interface operation

  The reduced media independent interface (RMII) is a low cost alternative to the IEEE 802.3 MII standard. This interface provides the functionality of the MII interface on a total of 8 pins instead of 18. The RMII interface for 10/100 Ethernet MAC-PHY interface was defined by an industry consortium and is not currently included in the IEEE 802.3 standard. The RMII_MODE bit in ETH_R_CNTRL register controls this functionality. If this bit and MII_MODE are enabled, then

the RMII interface is enabled. The RMII_10T bit in ETH_R_CNTRL register determines the speed of operation. The reference clock for RMII is always 50 MHz, but this clock can be divided by 10 within the RMII_10T bit of ETH_R_CNTRL register to support 10 Mbit/s operation.The PHY must be configured accordingly.

- 10 Mbit/s 7-wire interface operation

  The FEC support 7-wire interface used by many 10Mbit/s Ethernet transceivers.The MII_MODE bit in the ETH_R_CNTRL register controls this functionality. If this bit is cleared, MII mode is disabled and the 10Mbit/s 7-wire mode is enabled.

- Address recognition options

  Refer to the ETH_R_CNTRL register for address recognition options. Also, refer to Section 14.6.3, “Ethernet Address Recognition,” for a detailed description. The options supported are promiscuous, broadcast reject, individual address hash, or exact match and multicast hash match.

- Internal loopback

  Internal loopback mode is selected via the LOOP bit in the ETH_R_CNTRL register. Also, refer to Section 14.6.7, “MII Internal and External Loopback,” for a detailed description.

14.2 External Signal Description (Off Chip)

Table 14-1 describes the various FEC signals, as well as indicating which signals work in available modes.FEC*n* refers to FEC1 when *n* = 1 and FEC2 when *n* = 2.

Table 14-1. FEC signal Descriptions

Signal Name	MII	7-wire	RMII	I/O	Description
FEC*n*_COL	X	X	—	I	Asserted—This signal is asserted upon detection of a collision and remains asserted while the collision persists. The behavior of this signal is not specified when in full-duplex mode.
FEC*n*_CRS	X	—	—	I	Asserted—This signal is asserted when the transmit or receive medium is not idle. If a collision occurs, CRS remains asserted through the duration of the collision. In RMII mode, this signal is presented on the FEC*n*_RX_DV pin.
FEC*n*_MDC	X	—	X	O	Asserted—This signal provides a timing reference to the PHY for data transfers on the FEC*n*_MDIO signal. FEC*n*_MDC is a periodic and has no maximum high or low times.
FEC*n*_MDIO	X	—	X	I/O	Asserted—This signal transfers control/status information between the PHY and MAC. It transitions synchronously to FEC*n*_MDC. The FEC*n*_MDIO pin is a bidirectional pin. When the FEC operates in 10 Mbit/s 7-wire interface mode, this signal should be connected to V_{SS}.
FEC*n*_RX_CLK	X	X	—	I	Asserted—A continuous clock that provides a timing reference for FEC*n*_RX_DV, FEC*n*_RXD, and FEC*n*_RX_ER
FEC*n*_RX_DV/ RMII_CRS_DV	X	X	X	I	Asserted—When this signal is asserted, the PHY indicates a valid nibble is present on the MII. This signal remains asserted from the first recovered nibble of the frame through the last nibble. Assertion of FEC*n*_RX_DV must start no later than the SFD and excludes any EOF. In RMII mode, this pin also generates FEC*n*_CRS signal.

Table 14-1. FEC signal Descriptions (Continued)

Signal Name	MII	7-wire	RMII	I/O	Description
FEC*n*_RXD[3:2]	X	—	—	I	Asserted—These signals contain the Ethernet input data transferred from PHY to the MAC when FEC*n*_RX_DV is asserted.
FEC*n*_RXD_1/ RMII_RX1	X	—	X	I	Asserted—This signal contains the Ethernet input data transferred from PHY to the MAC when FEC*n*_RX_DV is asserted.
FEC*n*_RXD_0/ RMII_RX0	X	X	X	I	Asserted—This signal contains the Ethernet input data transferred from PHY to the MAC when FEC*n*_RX_DV is asserted.
FEC*n*_RX_ER/ RMII_RX_ER	X	—	X	I	Asserted—When asserted with FEC*n*_RX_DV, the PHY has detected an error in the current frame. When FEC*n*_RX_DV is not asserted, RX_ER has no effect.
FEC*n*_TX_CLK/ RMII_REF_CLK	X	X	X	I	Asserted—A continuous clock that provides a timing reference for FEC*n*_TX_EN, FEC*n*_TXD, and FEC*n*_TX_ER. In RMII mode, this signal is the reference clock for receive, transmit, and the control interface.
FEC*n*_TXD[3:2]	X	—	—	O	Asserted— These signals contain the serial output Ethernet data and valid only during assertion of FEC*n*_TX_EN
FEC*n*_TXD_1/ RMII_TX1	X	—	X	O	Asserted— This signal contains the serial output Ethernet data and valid only during assertion of FEC*n*_TX_EN
FEC*n*_TXD_0/ RMII_TX0	X	X	X	O	Asserted— This signal contains the serial output Ethernet data and valid only during assertion of FEC*n*_TX_EN
FEC*n*_TX_EN/ RMII_TX_EN	X	X	X	O	Asserted—Assertion of this signal indicates there are valid nibbles being presented on the MII. This signal is asserted with the first nibble of the preamble and is negated prior to the first FEC*n*_TX_CLK following the final nibble of the frame.
FEC*n*_TX_ER	X	—	—	O	Asserted—Assertion of this signal for one or more clock cycles while FEC*n*_TX_EN is also asserted, PHY sends one or more illegal symbols. FEC*n*_TX_ER has no effect when operating at 10 Mbit/s or when FEC*n*_TX_EN is deasserted.

14.3 Memory Map and Register Definition

14.3.1 Overview

There are two FECs (FEC1, FEC2) in the MPC5125. Each FEC is programmed by a combination of control/status registers (CSRs) and buffer descriptors. The CSRs are used for mode control, interrupts, and to extract status information. The descriptors are used to pass data buffers and related buffer or frame information between the hardware and software.

All accesses to and from the registers must be via 32-bit accesses. There is no support for accesses other than 32-bit.

This section defines the memory map and the registers, and then defines the buffer descriptors.

14.3.2 Top-Level Module Memory Map

Each FEC implementation requires a 1 KB memory map space that is divided into two sections of 512 bytes each. The first is used for control/status registers. The second contains event/statistic counters held in the MIB block. Table 14-2 defines the top-level memory map for each FEC.

Table 14-2. Module Memory Map

Address	Function
0x000–1FF	Control/status registers
0x200–3FF	MIB block counters

14.3.3 Detailed Memory Map—Control/Status Registers

For each FEC, Table 14-3 shows the address of the register, what block the register pertains to, the name of the register, and a brief description of the register.

Table 14-3. FEC memory map

Offset from FEC_BASE[1] FEC1: 0xFF40_2800 FEC2: 0xFF40_4800	Register	Access[2]	Reset Value[3]	Section/Page
0x000	ETH_FEC_ID—FEC Identification Register	R/W	0x0000_0000	14.3.5.1/14-343
0x004	ETH_IEVENT—Interrupt Event Register	R/W	0x0000_0000	14.3.5.2/14-344
0x008	ETH_IMASK—Interrupt Mask Register	R/W	0x0000_0000	14.3.5.3/14-346
0x00C	Reserved			
0x010	ETH_R_DES_ACTIVE—CSR Receive Descriptor Active Register	R/W	0x0000_0000	14.3.5.4/14-347
0x014	ETH_X_DES_ACTIVE—CSR Transmit Descriptor Active Register	R/W	0x0000_0000	14.3.5.5/14-348
0x018–0x023	Reserved			
0x024	ETH_ECNTRL—Ethernet Control register	R/W	0x0000_0000	14.3.5.6/14-349
0x028–0x03C	Reserved			
0x040	ETH_MII_DATA—MII Management Frame Register	R/W	0x0000_0000	14.3.5.7/14-350
0x044	ETH_MII_SPEED—MII Speed Control Register	R/W	0x0000_0000	14.3.5.8/14-351
0x048–0x063	Reserved			
0x064	ETH_MIB_CONTROL—MIB Control Register	R/W	0xC000_0000	14.3.5.9/14-352
0x068–0x083	Reserved			
0x084	ETH_R_CNTRL—Receive Control Register	R/W	0x05EE_0001	14.3.5.10/14-353
0x088	ETH_R_HASH—Receive Hash Register	R		14.3.5.11/14-355
0x08C–0x0C3	Reserved			
0x0C4	ETH_X_CNTRL—Transmit Control Register	R/W	0x0000_0000	14.3.5.12/14-355
0x0C8–0x0E3	Reserved			

Table 14-3. FEC memory map (Continued)

Offset from FEC_BASE[1] FEC1: 0xFF40_2800 FEC2: 0xFF40_4800	Register	Access[2]	Reset Value[3]	Section/Page
0x0E4	ETH_PADDR1—Physical Address Low Register	R/W	0x0000_0000	14.3.5.13/14-356
0x0E8	ETH_PADDR2—Physical Address High Register	R/W	0x0000_8808	14.3.5.14/14-357
0x0EC	ETH_OP_PAUSE—Opcode/Pause Duration Register	R/W	0x0001_UUUU	14.3.5.15/14-358
0x0F0–0x117	Reserved			
0x118	ETH_IADDR1—Descriptor Individual Address 1	R/W	—[4]	14.3.5.16/14-358
0x11C	ETH_IADDR2—Descriptor Individual Address 2	R/W	—[4]	14.3.5.17/14-359
0x120	ETH_GADDR1—Descriptor Group Address 1	R/W	—[4]	14.3.5.18/14-359
0x124	ETH_GADDR2—Descriptor Group Address 2	R/W	—[4]	14.3.5.19/14-360
0x128–0x143	Reserved			
0x144	ETH_X_WMRK—FIFO Transmit FIFO Watermark	R/W	0x0000_0000	14.3.5.20/14-361
0x148	Reserved			
0x14C	ETH_R_BOUND—FIFO Receive Bound Register	R	0x0000_0600	14.3.5.21/14-361
0x150	ETH_R_FSTART—FIFO Receive Start Register	R/W	0x0000_0500	14.3.5.22/14-362
0x154–0x17F	Reserved			
0x180	ETH_R_DES_START—Beginning of Receive Descriptor Ring	R/W	—[4]	14.3.5.23/14-363
0x184	ETH_X_DES_START—Beginning of Transmit Descriptor Ring	R/W	—[4]	14.3.5.24/14-363
0x188	ETH_R_BUFF_SIZE—Receive Buffer Size Register	R/W	—[4]	14.3.5.25/14-364
0x18C–0x1F0	Reserved			
0x1F4	ETH_DMA_CONTROL—DMA Function Control Register	R/W	0xU000_0000	14.3.5.26/14-364
0x1F8–0x1FF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

[3] In this column, the symbol “U” indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

[4] Reset value is indeterminate.

14.3.4 MIB Block Counters Memory Map

Table 14-4 defines the MIB counters memory map, which defines the locations in the MIB RAM space where hardware-maintained counters reside. These fall in the 0x0200–0x03FF address offset range. The counters are divided into two groups.

RMON counters cover the Ethernet statistics counters defined in RFC 1757. In addition to the counters defined in the Ethernet statistics group, a counter is included to count truncated frames because the FEC supports frame lengths only as large as 2047 bytes. The RMON counters are implemented independently for transmit and receive to ensure accurate network statistics when operating in full-duplex mode.

IEEE counters support the mandatory and recommended counter packages defined in section 5 of ANSI/IEEE Std. 802.3 (1998 edition). The IEEE basic package objects are supported by the FEC, but do not require counters in the MIB block. In addition, some of the recommended package objects that are supported do not require MIB counters. Counters for transmit and receive full-duplex flow control frames are also included.

Table 14-4. MIB Counters

Offset from FEC_BASE[1] FEC1: 0xFF40_2800 FEC2: 0xFF40_4800	Register	Access	Reset Value	Section/Page
0x200	RMON_T_DROP—Frames counted incorrectly			
0x204	RMON_T_PACKETS—RMON TX packet count			
0x208	RMON_T_BC_PKT—RMON TX broadcast packets			
0x20C	RMON_T_MC_PKT—RMON TX multicast packets			
0x210	RMON_T_CRC_ALIGN—RMON TX packets with CRC/align error			
0x214	RMON_T_UNDERSIZE—RMON TX packets < 64 bytes; good CRC			
0x218	RMON_T_OVERSIZE—RMON TX packets > MAX_FL bytes; good CRC			
0x21C	RMON_T_FRAG—RMON TX packets < 64 bytes; bad CRC			
0x220	RMON_T_JAB—RMON TX packets > MAX_FL bytes; bad CRC			
0x224	RMON_T_COL—RMON TX collision count			
0x228	RMON_T_P64—RMON TX 64-byte packets			
0x22C	RMON_T_P65TO127—RMON TX 65- to 127-byte packets			
0x230	RMON_T_P128TO255—RMON TX 128- to 255-byte packets			

Table 14-4. MIB Counters (Continued)

Offset from FEC_BASE[1] FEC1: 0xFF40_2800 FEC2: 0xFF40_4800	Register	Access	Reset Value	Section/Page
0x234	RMON_T_P256TO511—RMON TX 256- to 511-byte packets			
0x238	RMON_T_P512TO1023—RMON TX 512- to 1023-byte packets			
0x23C	RMON_T_P1024TO2047—RMON TX 1024- to 2047-byte packets			
0x240	RMON_T_P_GTE2048—RMON TX packets with > 2048 bytes			
0x244	RMON_T_OCTETS—RMON TX octets			
0x248	IEEE_T_DROP—Frames counted incorrectly			
0x24C	IEEE_T_FRAME_OK—Frames transmitted OK			
0x250	IEEE_T_1COL—Frames transmitted with single collision			
0x254	IEEE_T_MCOL—Frames transmitted with multiple collisions			
0x258	IEEE_T_DEF—Frames transmitted after deferral delay			
0x25C	IEEE_T_LCOL—Frames transmitted with late collision			
0x260	IEEE_T_EXCOL—Frames transmitted with excessive collisions			
0x264	IEEE_T_MACERR—Frames transmitted with TX FIFO underrun			
0x268	IEEE_T_CSERR—Frames transmitted with carrier sense error			
0x26C	IEEE_T_SQE—Frames transmitted with SQE error			
0x270	T_FDXFC—Flow control pause frames transmitted			
0x274	IEEE_T_OCTETS_OK—Octet count for frames transmitted without error			
0x278–0x27C	Reserved			
0x280	RMON_R_DROP—Frames counted incorrectly			
0x284	RMON_R_PACKETS—RMON RX packet count			
0x288	RMON_R_BC_PKT—RMON RX broadcast packets			
0x28C	RMON_R_MC_PKT—RMON RX multicast packets			
0x290	RMON_R_CRC_ALIGN—RMON RX packets with CRC/align error			
0x294	RMON_R_UNDERSIZE—RMON RX packets < 64 bytes; good CRC			
0x298	RMON_R_OVERSIZE—RMON RX packets > MAX_FL bytes; good CRC			
0x29C	RMON_R_FRAG—RMON RX packets < 64 bytes; bad CRC			
0x2A0	RMON_R_JAB—RMON RX packets > MAX_FL bytes; bad CRC			

Table 14-4. MIB Counters (Continued)

Offset from FEC_BASE[1] FEC1: 0xFF40_2800 FEC2: 0xFF40_4800	Register	Access	Reset Value	Section/Page
0x2A4	RMON_R_RESVD_0—			
0x2A8	RMON_R_P64—RMON RX 64-byte packets			
0x2AC	RMON_R_P65TO127—RMON RX 65- to 127-byte packets			
0x2B0	RMON_R_P128TO255—RMON RX 128- to 255-byte packets			
0x2B4	RMON_R_P256TO511—RMON RX 256- to 511-byte packets			
0x2B8	RMON_R_P512TO1023—RMON RX 512- to 1023-byte packets			
0x2BC	RMON_R_P1024TO2047—RMON RX 1024- to 2047-byte packets			
0x2C0	RMON_R_P_GTE2048—RMON RX packets with > 2048 bytes			
0x2C4	RMON_R_OCTETS—RMON RX octets			
0x2C8	IEEE_R_DROP—Frames counted incorrectly			
0x2CC	IEEE_R_FRAME_OK—Frames received OK			
0x2D0	IEEE_R_CRC—Frames received with CRC error			
0x2D4	IEEE_R_ALIGN—Frames received with alignment error			
0x2D8	IEEE_R_MACERR—Receive FIFO overflow count			
0x2DC	R_FDXFC—Flow control pause frames received			
0x2E0	IEEE_R_OCTETS_OK—Octet count for frames received without error			
0x2E4–0x3FF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

14.3.5 Register Descriptions

14.3.5.1 FEC ID (ETH_FEC_ID) Register

The FEC ID (ETH_FEC_ID) register is a read-only register. The FEC ID register is used to identify the FEC block and revision.

Address: Base + 0x000 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FEC_ID															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	DMA	FIFO	SMII	FEC_REV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-2. FEC ID Register (ETH_FEC_ID)

Table 14-5. ETH_FEC_ID field descriptions

Field	Description
FEC_ID	Unique identifier for FEC. 0000
DMA	DMA function included in the FEC 0 EC does not include DMA. 1 FEC includes DMA (ETH_DMA_CONTROL register contains DMA revision).
FIFO	FIFO function included in the FEC 0 FEC does not include a FIFO. 1 FEC includes a FIFO (FIFO_ID register contains the FIFO revision).
SMII	The Ethernet PHY interface configuration. 1 SMII (serial MII) interface. This 6-pin serial MII option requires that the MII_MODE bit of the ETH_R_CNTRL register is set equal to 1. 0 MII (18 pins) or 7-wire (select MII or 7-wire via the MII_MODE bit of the ETH_R_CNTRL register).
FEC_REV	Value identifies the revision of the FEC. 00 Initial revision.

14.3.5.2 Interrupt Event Register (ETH_IEVENT)

When an event occurs that sets a bit in the Interrupt Event (ETH_IEVENT) register, an interrupt is generated if the corresponding bit in the Interrupt Enable (ETH_IMASK) register is also set. The bit in the interrupt event register is cleared if a 1 is written to that bit position. Writing 0 has no effect. This register is cleared upon hardware reset.

These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts that may occur in normal operation are GRA, TFINT, TXB, RFINT, RXB, and MII. Interrupts resulting from errors/problems detected in the network or transceiver are HBERR, BABR, BABT, LATE_COL and COL_RETRY_LIM. Interrupts resulting from internal errors are EBERR and XFIFO_UN.

Some of the error interrupts are independently counted in the MIB block counters. Software may choose to mask off these interrupts because these errors are visible to network management via the MIB counters.

- HBERR—ieee_t_sqe
- BABR—rmon_r_oversize (good CRC), rmon_r_jab (bad CRC)
- BABT—rmon_t_oversize (good CRC), rmon_t_jab (bad CRC)
- LATE_COL—ieee_t_lcol
- COL_RETRY_LIM—ieee_t_excol
- XFIFO_UN—ieee_t_macerr

Address: Base + 0x004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HB ERR	BABR	BABT	GRA	TFINT	TXB	RF INT	RXB	MII	EB ERR	LATE _COL	COL_ RETR Y_LIM	XFIF O_UN	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-3. Interrupt Event Register (ETH_IEVENT)

Table 14-6. ETH_IEVENT field descriptions

Field	Description
HBERR	Heartbeat error. This interrupt indicates HBC is set in the ETH_X_CNTRL register and the COL input was not asserted within the heartbeat window following a transmission.
BABR	Babbling receive error. This bit indicates a frame was received with length in excess of ETH_R_CNTRL[MAX_FL] bytes.
BABT	Babbling transmit error. This bit indicates the transmitted frame length has exceeded ETH_R_CNTRL[MAX_FL] bytes. This condition is usually caused by a frame that is too long being placed into the transmit data buffer(s). Truncation does not occur.
GRA	Graceful stop complete. This interrupt is asserted for one of three reasons. Graceful stop means that the transmitter is put into a pause state after completion of the frame currently being transmitted. • A graceful stop initiated by setting the GTS bit of the ETH_X_CNTRL register is now complete. • A graceful stop initiated by setting the FC_PAUSE bit of the ETH_X_CNTRL register is now complete. • A graceful stop initiated by the reception of a valid full-duplex flow control pause frame is now complete. Refer to Section 14.6.4, “Full-Duplex Flow Control.”
TFINT	Transmit frame interrupt. This bit indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.
TXB	Transmit buffer interrupt. This bit indicates a transmit buffer descriptor that had the R bit set in its status word has been updated.
RFINT	Receive frame interrupt. This bit indicates a frame has been received and the last corresponding buffer descriptor has been updated.
RXB	Receive buffer interrupt. This bit indicates a receive buffer descriptor that had the E bit set in its status word has been updated.

Table 14-6. ETH_IEVENT field descriptions (Continued)

Field	Description
MII	MII interrupt. This bit indicates the MII has completed the requested data transfer.
EBERR	Ethernet bus error. This bit indicates a system bus error occurred when a DMA transaction was underway. When the EBERR bit is set, the ETHER_EN of the of the ETH_ECNTRL register is cleared, halting frame processing by the FEC.
LATE_COL	Late collision. This bit indicates a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame is truncated with a bad CRC and the remainder of the frame is discarded.
COL_RETRY_LIM	Collision retry limit. This bit indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This situation can only occur in half-duplex mode.
XFIFO_UN	Transmit FIFO underrun. This bit indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.

14.3.5.3 Interrupt Mask (ETH_IMASK) Register

The Interrupt Mask (ETH_IMASK) register provides control over which possible interrupt events are allowed to generate an actual interrupt. All implemented bits in this CSR are read/write. This register is cleared upon a hardware reset. If the corresponding bits in both the ETH_IEVENT and ETH_IMASK registers are set, the interrupt is signalled to the CPU. The interrupt signal remains asserted until a 1 is written to the ETH_IEVENT bit (write 1 to clear) or a 0 is written to the ETH_IMASK bit.

Address: Base + 0x008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBEEN	BREN	BTEN	GRAEN	TFIEN	TBIEN	RFIEN	RBIEN	MIIEN	EBERREN	LCEN	CRLEN	XFUNEN	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0[1]	0[1]	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-4. Interrupt Mask Register (ETH_IMASK)

[1] Reserved. Do not write. This bit must remain cleared.

Table 14-7. ETH_IMASK field descriptions

Field	Description
HBEEN	Heartbeat error interrupt enable
BREN	Babbling receiver interrupt enable
BTEN	Babbling transmitter interrupt enable
GRAEN	Graceful stop interrupt enable
TFIEN	Transmit frame interrupt enable
TBIEN	Transmit buffer interrupt enable
RFIEN	Receive frame interrupt enable
RBIEN	Receive buffer interrupt enable
MIIEN	MII interrupt enable
EBERREN	Ethernet controller bus error enable
LCEN	Late collision enable
CRLEN	Collision retry limit enable
XFUNEN	Transmit FIFO underrun enable

14.3.5.4 CSR Receive Descriptor Active (ETH_R_DES_ACTIVE) Register

The CSR Receive Descriptor Active (ETH_R_DES_ACTIVE) register is a command register that should be written to indicate the receive descriptor ring has been updated (empty receive buffers have been produced by the driver with the E bit set).

The R_DES_ACTIVE bit in the ETH_R_DES_ACTIVE register is set when the register is written. This is independent of the data actually written. When set, the FEC polls the receive descriptor ring and processes receive frames, provided ETH_ECNTRL[ETHER_EN] is also set. After the FEC polls a receive descriptor whose ownership bit is not set, the FEC clears the R_DES_ACTIVE bit and ceases receive descriptor ring polling until the bit is set again, signifying additional descriptors have been placed into the receive descriptor ring.

The ETH_R_DES_ACTIVE register is cleared at reset and by the clearing of the ETHER_EN bit of the ETH_ECNTRL register.

Address: Base + 0x010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	R_DES_ACTIVE	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-5. CSR Receive Descriptor Active (ETH_R_DES_ACTIVE) Register

Table 14-8. ETH_R_DES_ACTIVE field descriptions

Field	Description
R_DES_ACTIVE	This bit is set to 1 when this register is written, regardless of the value written. It is cleared by the FEC device when no additional ready descriptors remain in the receive ring.

14.3.5.5 CSR Transmit Descriptor Active (ETH_X_DES_ACTIVE) Register

The CSR Transmit Descriptor Active (ETH_X_DES_ACTIVE) register is a command register that should be written to indicate the transmit descriptor ring has been updated (transmit buffers have been produced by the driver with the R bit set in the buffer descriptor).

The X_DES_ACTIVE bit in the ETH_X_DES_ACTIVE register is set when the register is written. This is independent of the data actually written. When set, the FEC polls the transmit descriptor ring and process transmit frames, provided ETHER_EN bit of the ETH_ECNTRL register is also set. After the FEC polls a transmit descriptor whose ownership bit is not set, the FEC clears the X_DES_ACTIVE bit and ceases transmit descriptor ring polling until the bit is set again, signifying additional descriptors have been placed into the transmit descriptor ring.

The ETH_X_DES_ACTIVE register is cleared at reset and by the clearing of the ETHER_EN bit of the ETH_ECNTRL register.

Address: Base + 0x014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	X_DES_ACTIVE	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-6. CSR Transmit Descriptor Active (ETH_X_DES_ACTIVE) Register

Table 14-9. ETH_X_DES_ACTIVE field descriptions

Field	Description
X_DES_ACTIVE	This bit is set to 1 when this register is written, regardless of the value written. It is cleared by the FEC device when no additional ready descriptors remain in the transmit ring.

14.3.5.6 Ethernet Control (ETH_ECNTRL) Register

The Ethernet Control (ETH_ECNTRL) register is a read/write user register; however, some fields may be altered by hardware as well. The ETH_ECNTRL register enables/disables the FEC. The Reserved bits must be cleared.

Address: Base + 0x024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1	1	1	1	0	0[1]	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ETHER_EN	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0[1]	0	0

Figure 14-7. Ethernet Control (ETH_ECNTRL) Register

[1] Reserved. Do not write. This bit must remain cleared.

Table 14-10. ETH_ECNTRL field descriptions

Field	Description
ETHER_EN	Ethernet enable. When this bit is set, the fast Ethernet controller is enabled, and reception and transmission is possible. When this bit is cleared, reception is immediately stopped and transmission is stopped after a bad CRC is appended to any frame currently being transmitted. The buffer descriptor(s) for an aborted transmit frame are not updated following deassertion of ETHER_EN. When ETHER_EN is deasserted, the DMA, buffer descriptor, and FIFO control logic are reset, including buffer descriptor and FIFO pointers. The ETHER_EN bit is altered by hardware under the following conditions: • If ETH_ECNTRL[RESET] is written to a 1 by software, ETHER_EN is cleared • If error conditions occur, causing the EBERR bit of the ETH_IEVENT register to set, ETHER_EN is cleared.
RESET	Ethernet controller reset. When this bit is set, the equivalent of a hardware reset is performed, but it is local to the FEC. The ETHER_EN bit is cleared and all other FEC registers take their reset values. Also, any transmission/reception currently in progress is abruptly aborted. This bit is automatically cleared by hardware during the reset sequence. The reset sequence takes approximately 8 clock cycles after RESET is written with a 1. It's recommended to read ETH_ECNTRL register back and this can give enough time to go through reset sequence

14.3.5.7 MII Management Frame (ETH_MII_DATA) Register

The MII Management Frame (ETH_MII_DATA) register does not reset to a defined value. The ETH_MII_DATA register communicates with the attached MII-compatible PHY device, providing read/write access to MII registers. Performing a write to the ETH_MII_DATA register causes a management frame to be sourced, unless the ETH_MII_SPEED register has been programmed to 0. Writing to ETH_MII_DATA when ETH_MII_SPEED equals 0, and the ETH_MII_SPEED register is then written to a non-zero value, an MII frame is generated with the data previously written to the ETH_MII_DATA register. This allows ETH_MII_DATA and ETH_MII_SPEED to be programmed in either order if ETH_MII_SPEED is currently zero.

Address: Base + 0x040 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ST				OP				PA				RA			TA
Reset[1]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 14-8. MII Management Frame (ETH_MII_DATA) Register

[1] This register is undefined at reset.

Table 14-11. ETH_MII_DATA field descriptions

Field	Description
ST	Start of frame delimiter. These bits must be programmed to 0b01 for a valid MII management frame.

Table 14-11. ETH_MII_DATA field descriptions (Continued)

Field	Description
OP	Operation code. This field must be programmed to 0b10 (read) or 0b01(write) to generate a valid MII management frame. A value of 11 produces read frame operation, while a value of 00 produces write frame operation, but these frames are not MII-compliant.
PA	PHY address. This field specifies one of as many as 32 attached PHY devices.
RA	Register address. This field specifies one of as many as 32 registers within the specified PHY device.
TA	Turn around. This field must be programmed to 0b10 to generate a valid MII management frame.
DATA	Management frame data. This is the field for data to be written to or read from the PHY register.

Write the ETH_MII_DATA register to perform a read or write operation on the MII management interface. To generate a valid read or write management frame, the ST field must be written with a 01 pattern, the OP field must be written with a 01 (management register write frame) or 10 (management register read frame), and the TA field must be written with a 10. If other patterns are written to these fields, a frame is generated, but does not comply with the IEEE 802.3 MII definition. When op field equals 1x, it produces a read-frame operation, while op equals 0x produces a write-frame operation.

To generate an 802.3-compliant MII management interface write frame (write to a PHY register), write {01 01 PHYAD REGAD 10 DATA} to the ETH_MII_DATA register. Writing this pattern causes the control logic to shift out the data in the ETH_MII_DATA register following a preamble generated by the control state machine. During this time, the contents of the ETH_MII_DATA register is altered as the contents are serially shifted, and are unpredictable if read by the user. The MII interrupt is generated after the write management frame operation has been completed. At this time, the contents of the ETH_MII_DATA register match the original value written.

To generate an MII management interface read frame (read a PHY register), write {01 10 PHYAD REGAD 10 XXXX} to the ETH_MII_DATA register (the content of the data field is a “don’t care”). Writing this pattern causes the control logic to shift out the data in the ETH_MII_DATA register following a preamble generated by the control state machine. During this time, the contents of the ETH_MII_DATA register are altered as the contents are serially shifted and are unpredictable if read. The MII interrupt is generated after the read management frame operation has completed. At this time, the contents of the ETH_MII_DATA register match the original value written, except for the data field, where contents have been replaced by the value read from the PHY register.

If the MII_DATA register is written while frame generation is in progress, the frame contents are altered. Software should use the MII_STATUS register and/or the MII interrupt to avoid writing to the ETH_MII_DATA register while frame generation is in progress.

14.3.5.8 MII Speed Control (ETH_MII_SPEED) Register

The MII Speed Control (ETH_MII_SPEED) register provides control of the MII clock (MDC pin) frequency, allows dropping the preamble on the MII management frame, and provides observability (intended for manufacturing test) of an internal counter used in generating the MDC clock signal.

Address: Base + 0x044 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	DIS_PREAMBLE	MII_SPEED						0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-9. MII Speed Control (ETH_MII_SPEED) Register

Table 14-12. ETH_MII_SPEED field descriptions

Field	Description
DIS_PREAMBLE	Asserting this bit causes preamble (32 ones) not to be prepended to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY device(s) does not require it.
MII_SPEED	MII_SPEED controls the frequency of the MII management interface clock (MDC) relative to system clock. A value of 0 in this field turns off the MDC and leaves it in a low-voltage state. Any non-zero value results in the MDC frequency of 1/(mii_speed × 2) of the system clock frequency.

To be compliant with the IEEE MII specification, the MII_SPEED field must be programmed with a value that provides an MDC frequency of less than or equal to 2.5 MHz. The MII_SPEED bitfield must be set to a non-zero value to source a read or write management frame. After the management frame is complete, the ETH_MII_SPEED register may optionally be set to 0 to turn off the MDC. The MDC generated has a 50% duty cycle, except when the MII_SPEED bitfield is changed during operation (the change takes effect following a rising or falling edge of MDC).

If the system clock is 25 MHz, programming the ETH_MII_SPEED register to 0x0000_0005 results in an MDC frequency of 25 MHz × 1/(2 × 5) = 2.5 MHz. Table 14-13 shows optimum values for the MII_SPEED bitfield as a function of system clock frequency.

Table 14-13. Programming Examples for MII_SPEED Bitfield

IPS Clock Frequency	MII_SPEED bitfield	MDC Frequency
25 MHz	0x05	2.5 MHz
33 MHz	0x07	2.36 MHz
40 MHz	0x08	2.5 MHz
50 MHz	0x0A	2.5 MHz
66 MHz	0x0E	2.36 MHz
83 MHz	0x11	2.44 MHz

14.3.5.9 MIB Control (ETH_MIB_CONTROL) Register

The MIB Control (ETH_MIB_CONTROL) register is a read/write register that provides control of and observes the state of the MIB block. If it is necessary to disable the MIB block operation, this register is

accessed by user software. For example, to clear all MIB counters in RAM, disable the MIB block, clear all the MIB RAM locations, and then enable the MIB block. The MIB_DISABLE bit is reset to 1.

Address: Base + 0x064 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIB_DIS-ABLE	MIB_IDLE	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-10. MIB Control (ETH_MIB_CONTROL) Register

Table 14-14. ETH_MIB_CONTROL field descriptions

Field	Description
MIB_DISABLE	This is a read/write control bit. If set, the MIB logic halts and does not update any MIB counters.
MIB_IDLE	This is a read-only status bit. If set, the MIB block is not currently updating any MIB counters.

14.3.5.10 Receive Control (ETH_R_CNTRL) Register

The Receive Control (ETH_R_CNTRL) register controls the operational mode of the receive block and should only be written when the ETHER_EN bit of the ETH_ECNTRL register equals 0 (initialization time).

Address: Base + 0x084 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	MAX_FL										
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	0	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	RMII_ECHO	RMII_LOOP	RMII_10T	RMII_MODE	0	0	FCE	BC_REJ	PROM	MII_MODE	DRT	LOOP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 14-11. Receive Control (ETH_R_CNTRL) Register

Table 14-15. ETH_R_CNTRL field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>MAX_FL</td><td>Maximum frame length. This is a user read/write field that resets to 0x5EE (decimal 1518). Length is measured starting at DA (destination address) and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL causes a BABT interrupt to occur. Receive frames longer than MAX_FL cause a BABR interrupt to occur and set the LG bit in the end-of-frame buffer descriptor. The recommended values to be programmed are 1518 (the default) or 1522 (if VLAN tags are supported).</td></tr>
<tr><td>RMII_ECHO</td><td>RMII Echo. Enables RMII echo mode. RMII 2-bit receive data is processed through the RMII receive logic to the FEC and also routes through the RMII's transmit logic to become RMII 2-bit transmit data out.
0 Normal operation
1 RMII echo mode enabled.
Note: When RMII_ECHO is set, proper operation is guaranteed only when RMII_MODE = 1, RMII_LOOP = 0 and LOOP = 0.</td></tr>
<tr><td>RMII_LOOP</td><td>RMII loopback. Enables RMII loopback mode. Causes the MII transmit outputs from the Ethernet controller to loop back to the Ethernet controllers's MII receive inputs through the RMII transmit/receive logic.
0 Normal operation
1 RMII loopback mode enabled.
Note: When RMII_LOOP is set, proper operation is guaranteed only when RMII_MODE = 1, RMII_ECHO = 0, LOOP =0, and ETH_X_CNTRL[FDEN] = 1.</td></tr>
<tr><td>RMII_10T</td><td>RMII 10-Base. Enables 10 Mbit/s mode of the RMII. Determines the clock frequency of the clock source to the FEC logic to support 10/100Mbit/s operations.
0 100 Mbit/s operation. The 50 Mhz RMII reference clock on FECn_TX_CLK is sent to the RMII, while a divided-by-2 version (25 MHz) is sent to the FEC.
1 10Mbit/s operation. The 50MHz RMII reference clock on FECn_TX_CLK is divided by 10 (5 MHz) and sent to the RMII, while a divided-by-20 version (2.5 MHz) is sent to the FEC MII part.</td></tr>
<tr><td>RMII_MODE</td><td>RMII Mode. Indicates if the FEC is in RMII or MII/7-wire mode.
0 FEC configured for MII or 7-wire mode as indicated by the MII_MODE bit.
1 FEC configured for RMII operation, only if the MII_MODE bit is set.
The following table shows the valid settings for RMII_MODE and MII_MODE.
<table>
<tr><th>MII_MODE</th><th>RMII_MODE</th><th>Description</th></tr>
<tr><td>1</td><td>0</td><td>MII mode</td></tr>
<tr><td>1</td><td>1</td><td>RMII mode</td></tr>
<tr><td>0</td><td>0</td><td>7-wire</td></tr>
</table>
</td></tr>
<tr><td>FCE</td><td>Flow control enable. If asserted, the receiver detects pause frames. The transmitter stops transmitting data frames for a given duration when pause frames are detected.</td></tr>
<tr><td>BC_REJ</td><td>Broadcast frame reject. If asserted, frames with DA = FFFF_FFFF_FFFF are rejected unless the PROM bit is set. If BC_REJ and PROM equal 1 individually, frames with broadcast DA are accepted and the MISS (M) bit is set in the receive buffer descriptor.</td></tr>
<tr><td>PROM</td><td>Promiscuous mode. All frames are accepted, regardless of address matching.</td></tr>
<tr><td>MII_MODE</td><td>Selects external interface mode.
0 7-wire mode (used only for serial 10Mbit/s)
1 MII or RMII mode as indicated by the RMII_MODE bit</td></tr>
<tr><td>DRT</td><td>Disable receive on transmit.
0 Receive path operates independently of transmit (use for full-duplex or to monitor transmit activity in half-duplex mode).
1 Disable reception of frames while transmitting (normally used for half-duplex mode).</td></tr>
</table>

Table 14-15. ETH_R_CNTRL field descriptions (Continued)

Field	Description
LOOP	Internal loopback. If set, transmitted frames are looped back internal to the device and the transmit output signals are not asserted. The system clock is substituted for the TX_CLK when loop is asserted. DRT must be set to 0 when asserting loop.

14.3.5.11 Receive Hash (ETH_R_HASH) Register

The read-only Receive Hash (ETH_R_HASH) register provides address recognition information from the receive block about the frame currently being received. This field is read by the FEC. These bits provide the FEC with information used in the address recognition subroutine.

Address: Base + 0x088 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FCE_DC	MULT-CAST	HASH						0	0	0	0	0	0	0	0
W																
Reset																

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset																

Figure 14-12. Receive Hash (ETH_R_HASH) Register

Table 14-16. ETH_R_HASH field descriptions

Field	Description
FCE_DC	This is a read-only view of the FCE bit in the ETH_R_CNTRL register.
MULTCAST	This bit is set if the current receive frame contained a multicast destination address (the least significant bit of the DA was set). It is cleared if the current receive frame does not correspond to a multicast address.
HASH	Corresponds to the hash value of the current receive frame's destination address. The hash value is a 6-bit field extracted from the least significant portion of the CRC register.

NOTE

The FCE_DC, MULTCAST, and HASH bits are not affected by $\overline{\text{PORESET}}$ or $\overline{\text{HRESET}}$.

14.3.5.12 Transmit Control (ETH_X_CNTRL) Register

The Transmit Control (ETH_X_CNTRL) register is read/write and is written to configure the transmit block. This register is cleared at system reset. Bits FDEN and HBC should be modified only when the ETHER_EN bit of the ETH_ECNTRL register equals 0.

Address: Base + 0x0C4 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	RFC_ PAUSE	TFC_ PAUSE	FDEN	HBC	GST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-13. Transmit Control (ETH_X_CNTRL) Register

Table 14-17. ETH_X_CNTRL field descriptions

Field	Description
RFC_PAUSE	This read-only status bit is asserted when a full-duplex flow control pause frame has been received and the transmitter is paused for the duration defined in this pause frame. This bit automatically clears when the pause duration is complete.
TFC_PAUSE	This bit is asserted to transmit a pause frame. When this bit is set, the MAC stops transmission of data frames after the current transmission is complete. At this time, the GRA interrupt in the INTR_EVENT register is asserted. With transmission of data frames stopped, the MAC transmits a MAC control pause frame. Next, the MAC clears the TFC_PAUSE bit and resumes transmitting data frames. If the transmitter is paused due to assertion of GTS or reception of a pause frame, the MAC may continue to transmit a MAC control pause frame.
FDEN	Full-duplex enable. If set, frames are transmitted independent of carrier sense and collision inputs. This bit should be modified only when ETHER_EN bit of the ETH_ECNTRL register is deasserted.
HBC	Heartbeat control. If set, the heartbeat check is performed following the end of transmission and the HB bit in the status register is set if the collision input does not assert within the heartbeat window. This bit should be modified only when the ETHER_EN bit of the ETH_ECNTRL register is deasserted.
GTS	Graceful transmit stop. When this bit is set, the MAC stops transmission after completion of any frame currently being transmitted, and the GRA interrupt in the INTR_EVENT register is asserted. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission is complete, a restart can be accomplished by clearing the GTS bit. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS equals 1, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There may be old frames in the transmit FIFO that are transmitted when GTS is reasserted. To avoid this, deassert ETHER_EN bit of the ETH_ECNTRL register following the GRA interrupt.

14.3.5.13 Physical Address Low (ETH_PADDR1) Register

The Physical Address Low (ETH_PADDR1) register contains the lower 32 bits (bytes 0, 1, 2, 3) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in bytes 0–3 of the 6-byte source address field when transmitting pause frames. This register is not reset and must be initialized by the user.

Address: Base + 0x0E4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PADDR1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PADDR1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-14. Physical Address Low (ETH_PADDR1) Register

Table 14-18. ETH_PADDR1 field descriptions

Field	Description
PADDR1	This field comprises bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address to be used for exact match, and the source address field in pause frames.

14.3.5.14 Physical Address High (ETH_PADDR2) Register

The Physical Address High (ETH_PADDR2) register contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the DA field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the 6-byte source address field when transmitting pause frames. Bits 16:31 of ETH_PADDR2 contain a constant-type field (0x8808) used for transmission of pause frames. This register is not reset and bits 0:15 must be initialized.

Address: Base + 0x0E8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PADDR2															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TYPE															
W																
Reset	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

Figure 14-15. Physical Address High (ETH_PADDR2) Register

Table 14-19. ETH_PADDR2 field descriptions

Field	Description
PADDR2	This field comprises bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address to be used for an exact match, and the source address field in pause frames.
TYPE	This is the type field in pause frames. These 16 bits are a constant value of 0x8808.

14.3.5.15 Opcode/Pause Duration (ETH_OP_PAUSE) Register

The Opcode/Pause Duration (ETH_OP_PAUSE) register contains the 16-bit opcode and 16-bit pause duration fields used in transmission of a pause frame. The OPCODE field is a constant value, 0x0001. When another node detects a pause frame, that node pauses transmission for the duration specified in the pause duration field. This register is not reset and must be initialized.

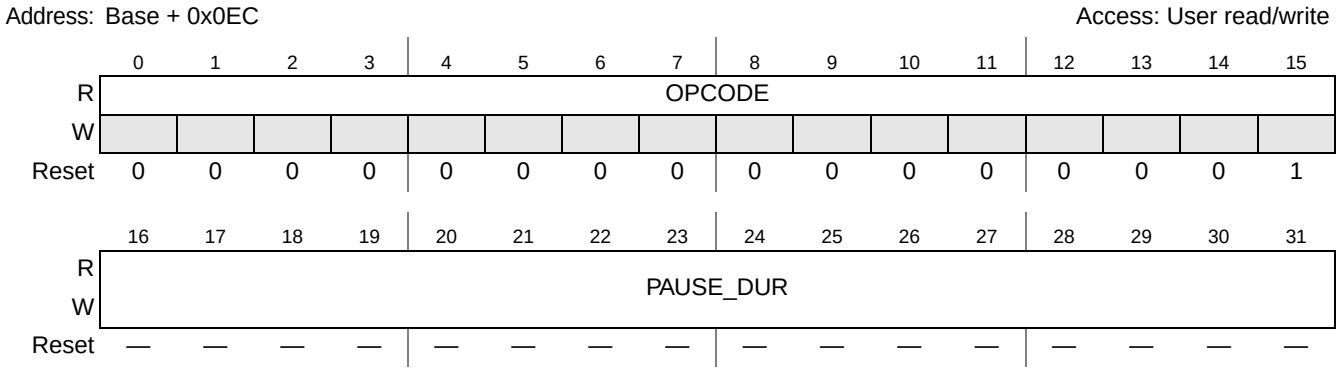


Figure 14-16. Opcode/Pause Duration (ETH_OP_PAUSE) Register

Table 14-20. ETH_OP_PAUSE field descriptions

Field	Description
OPCODE	This is the opcode field used in pause frames. These bits have a constant value of 0x0001.
PAUSE_DUR	This is the pause duration field used in pause frames.

14.3.5.16 Descriptor Individual Address 1 (ETH_IADDR1) Register

The Descriptor Individual Address 1 (ETH_IADDR1) register contains the upper 32 bits of the 64-bit individual address hash table used in the address recognition process to check for a possible match between the DA field of receive frames and an individual DA. This register is not reset and must be initialized.

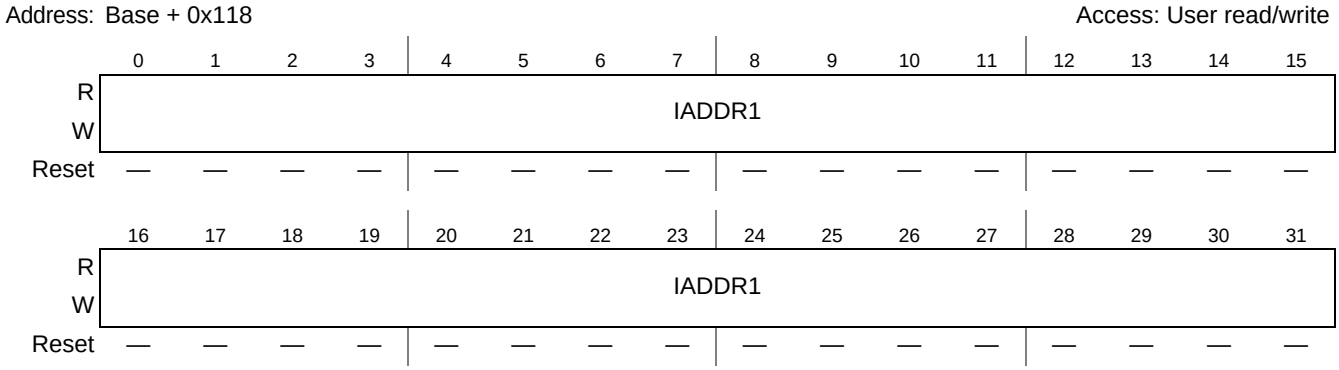


Figure 14-17. Descriptor Individual Address 1 (ETH_IADDR1) Register

Table 14-21. ETH_IADDR1 field descriptions

Field	Description
IADDR1	This field contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of ETH_IADDR1 contains hash index bit 63. Bit 0 of ETH_IADDR1 contains hash index bit 32.

14.3.5.17 Descriptor Individual Address 2 (ETH_IADDR2) Register

The Descriptor Individual Address 2 (ETH_IADDR2) register contains the lower 32 bits of the 64-bit individual address hash table used in the address recognition process to check for possible match between the DA field of receive frames and an individual DA. This register is not reset and must be initialized.

Address: Base + 0011C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IADDR2															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IADDR2															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 14-18. Descriptor Individual Address 2 (ETH_IADDR2) Register

Table 14-22. ETH_IADDR2 field descriptions

Field	Description
IADDR2	This field contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of ETH_IADDR2 contains hash index bit 32. Bit 0 of ETH_IADDR2 contains hash index bit 0.

14.3.5.18 Descriptor Group Address 1 (ETH_GADDR1) Register

The Descriptor Group Address 1 (ETH_GADDR1) register contains the upper 32 bits of the 64-bit hash address table used in the address recognition process for receive frames with a multicast address. This register is not affected by $\overline{\text{PORESET}}$ or $\overline{\text{HRESET}}$ and must be initialized.

Address: Base + 0x120 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	GADDR1															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GADDR1															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 14-19. Descriptor Group Address 1 (ETH_GADDR1) Register

Table 14-23. ETH_GADDR1 field descriptions

Field	Description
GADDR1	The ETH_GADDR1 register contains the upper 32 bits of the 64-bit hash table address used in the address recognition process for receive frames with a multicast address. Bit 31 of ETH_GADDR1 contains hash index bit 63. Bit 0 of ETH_GADDR1 contains hash index bit 32.

14.3.5.19 Descriptor Group Address 2 (ETH_GADDR2) Register

The Descriptor Group Address 2 (ETH_GADDR2) register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. This register is not affected by $\overline{\text{PORESET}}$ or $\overline{\text{HRESET}}$ and must be initialized.

Address: Base + 0x124 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	GADDR2															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GADDR2															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 14-20. Descriptor Group Address 2 (ETH_GADDR2) Register

Table 14-24. ETH_GADDR2 field descriptions

Field	Description
GADDR2	The ETH_GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of ETH_GADDR2 contains hash index bit 31. Bit 0 of ETH_GADDR2 contains hash index bit 0.

14.3.5.20 FIFO Transmit FIFO Watermark (ETH_X_WMRK) Register

The FIFO Transmit FIFO Watermark (ETH_X_WMRK) register is programmed to control the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows minimizing transmit latency (X_WMRK = 0X) or allows for larger bus access latency (X_WMRK = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the X_WMRK field may need to be modified to match a given system requirement (worst case bus access latency by the transmit data DMA channel).

Address: Base + 0x144 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X_WMRK	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-21. FIFO Transmit FIFO Watermark (ETH_X_WMRK) Register

Table 14-25. ETH_X_WMRK field descriptions

Field	Description
X_WMRK	Transmit FIFO watermark. Frame transmission begins when the number of bytes selected by this field have been written into the transmit FIFO if an end-of-frame has been written to the FIFO or if the FIFO is full before the selected number of bytes have been written. The options are: 0X 64 bytes written to xFIFO 10 128 bytes written to xFIFO 11 192 bytes written to xFIFO

14.3.5.21 FIFO Receive Bound(ETH_R_BOUND) Register

The FIFO Receive Bound (ETH_R_BOUND) register can be read to determine the upper address bound of the FIFO RAM. The highest address of FIFO_RAM is R_BOUND – 1.Drivers can use the value of R_BOUND, along with the ETH_R_FSTART[R_FSTART] register value, to appropriately divide the available FIFO RAM between the transmit and receive data paths.

The ETH_R_BOUND register is read-only.

Address: Base + 0x14C Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	1	R_BOUND								0	0
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 14-22. FIFO Receive Bound (ETH_R_BOUND) Register

Table 14-26. ETH_R_BOUND field descriptions

Field	Description
R_BOUND	Read-only. Determines the highest valid FIFO RAM address.

14.3.5.22 FIFO Receive Start (ETH_R_FSTART) Register

The FIFO Receive Start (ETH_R_FSTART) register is programmed to indicate the starting address of the receive FIFO. The transmit FIFO uses addresses from 0 to R_FSTART – 4. The receive FIFO uses addresses from R_FSTART to ETH_R_BOUND[R_BOUND] – 1, inclusive.

The ETH_R_FSTART register is initialized by hardware at reset. Write ETH_R_FSTART to change the default value.

Address: Base + 0x150 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	1	R_FSTART								0	0
W																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

Figure 14-23. FIFO Receive Start (ETH_R_FSTART) Register

Table 14-27. ETH_R_FSTART field descriptions

Field	Description
R_FSTART	This is the address of the first receive FIFO location. It acts as a delimiter between the receive and transmit FIFOs.

14.3.5.23 Beginning of Receive Descriptor Ring (ETH_R_DES_START) Register

The Beginning of Receive Descriptor Ring (ETH_R_DES_START) register is a pointer to the start of the circular receive buffer descriptor queue in external memory. This pointer must be 32-bit aligned; Write bits 30, 31 to 0. It is strongly recommended to be quad word-aligned (evenly divisible by 16) to get better system performance; write bits 28, 29, 30, and 31 to 0.

This register is not reset and must be initialized prior to operation.

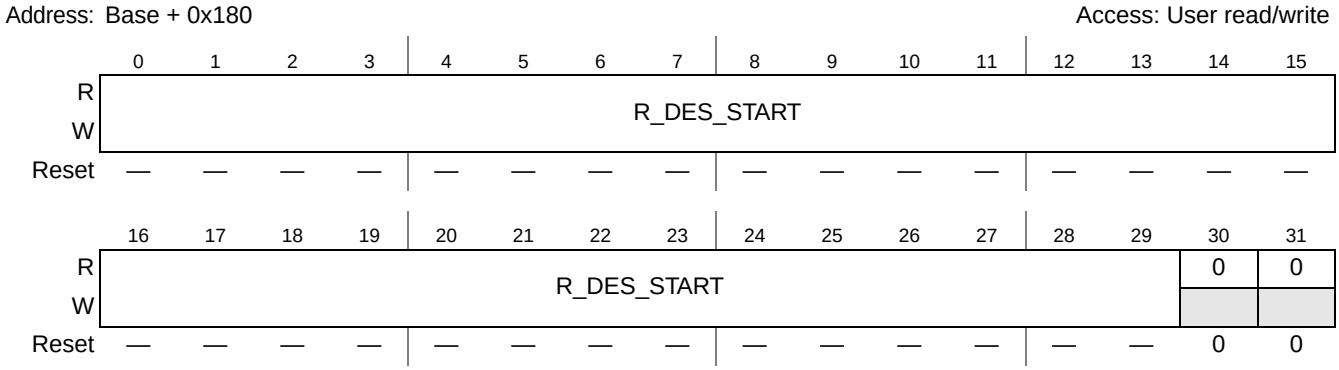


Figure 14-24. Beginning of Receive Descriptor Ring (ETH_R_DES_START) Register

Table 14-28. ETH_R_DES_START field descriptions

Field	Description
R_DES_START	This is a pointer to the start of the receive buffer descriptor queue.

14.3.5.24 Beginning of Transmit Descriptor Ring (ETH_X_DES_START) Register

The Beginning of Transmit Descriptor Ring (ETH_X_DES_START) register is a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be 32-bit aligned, write bits 30 and 31 to 0. It is strongly recommended to be quad word-aligned (evenly divisible by 16) to get better system performance; write bits 28, 29, 30, and 31 to 0.

This register is not reset and must be initialized prior to operation.

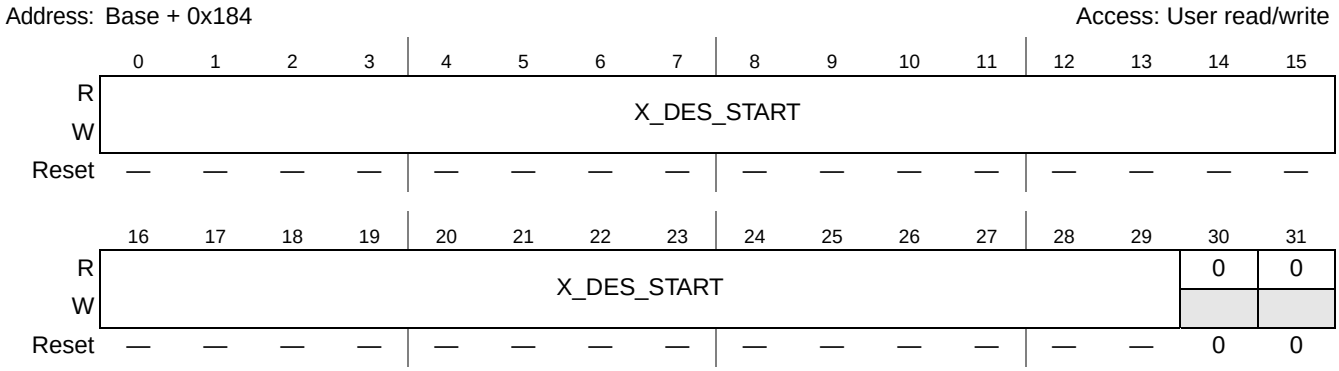


Figure 14-25. Beginning of Transmit Descriptor Ring (ETH_X_DES_START) Register

Table 14-29. ETH_X_DES_START field descriptions

Field	Description
X_DES_START	This is a pointer to the start of the transmit buffer descriptor queue.

14.3.5.25 Receive Buffer Size (ETH_R_BUFF_SIZE) Register

The Receive Buffer Size (ETH_R_BUFF_SIZE) register dictates the maximum size of all receive buffers. Only bits 21 – 27 are used, because receive frames are truncated at 2 KB – 1 bytes. This value should take into consideration that the receive CRC is always written into the last receive buffer. To allow one maximum-sized frame per buffer, the R_BUFF_SIZE bitfield must be set to ETH_R_CNTRL[MAX_FL] or larger. The value of the ETH_R_BUFF_SIZE register must be evenly divisible by 16. To ensure this, bits 3–0 are forced to 0. To minimize bus utilization (descriptor fetches), it is recommended that ETH_R_BUFF_SIZE be greater than or equal to 256 bytes.

The ETH_R_BUFF_SIZE register is not affected by reset and must be initialized.

Address: Base + 0x188 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	R_BUFF_SIZE							0	0	0	0
W																
Reset	0	0	0	0	0	—	—	—	—	—	—	—	—	—	—	0

Figure 14-26. Receive Buffer Size (ETH_R_BUFF_SIZE) Register

Table 14-30. ETH_R_BUFF_SIZE field descriptions

Field	Description
R_BUFF_SIZE	This is the receive buffer size.

14.3.5.26 DMA Function Control (ETH_DMA_CONTROL) Register

The DMA Function Control (ETH_DMA_CONTROL) register contains the function code and byte order fields used during each transfer between the DMA and the FEC interface. These bits can be written/read by the user. This register should be programmed only when the ETHER_EN bit of the ETH_ECNTRL register equals 0.

Address: Base + 0x1F4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DATA_BO	DESC_BO	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	—	—	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	DMA_REV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-27. DMA Function Control (ETH_DMA_CONTROL) Register

Table 14-31. ETH_DMA_CONTROL field descriptions

Field	Description
DATA_BO	The byte order control for data DMA transfers. 0 Little endian (bytes 0 and 3 swapped, bytes 1 and 2 swapped). 1 Big endian (no byte swapping). **Note:** This bit is not affected by reset and must be initialized before using the DMA controller.
DESC_BO	The byte order control for descriptor DMA transfers. 0 Little endian (bytes 0 and 3 swapped, bytes 1 and 2 swapped). 1 Big endian (no byte swapping). **Note:** The DATA_BO and DESC_BO fields are muxed to generate the IPM_BO signal on the master interface. **Note:** This bit is not affected by reset and must be initialized before using the DMA controller.
DMA_REV	DMA revision; read only

14.4 Initialization Information

This section describes which registers are reset due to hardware reset, which are reset by the FEC RISC (shown in Figure 14-1), and which locations must be initialized prior to enabling FEC.

14.4.1 Initialization Prior to Asserting ETHER_EN

Initialize portions of the FEC prior to setting the ETHER_EN bit of the ETH_ECNTRL register. The exact values depend on the particular application. The sequence is not important. Ethernet MAC registers requiring initialization are as follows:

- Start FEC Clock (SCCR1[FEC*n*_EN]). See Section 5.3.1.2, “System Clock Control Register 1 (SCCR1).”
- Initialize ETH_IMASK. See Section 14.3.5.3, “Interrupt Mask (ETH_IMASK) Register.”
- Clear ETH_IEVENT (write 0xFFFF_FFFF). See Section 14.3.5.2, “Interrupt Event Register (ETH_IEVENT).”

- ETH_X_WMRK (optional). See Section 14.3.5.20, “FIFO Transmit FIFO Watermark (ETH_X_WMRK) Register.”
- ETH_IADDR2/ETH_IADDR1 (see Section 14.3.5.16, “Descriptor Individual Address 1 (ETH_IADDR1) Register” and Section 14.3.5.17, “Descriptor Individual Address 2 (ETH_IADDR2) Register.”
- ETH_GADDR1/ETH_GADDR2. See Section 14.3.5.18, “Descriptor Group Address 1 (ETH_GADDR1) Register” and Section 14.3.5.19, “Descriptor Group Address 2 (ETH_GADDR2) Register.”
- ETH_PADDR1/ETH_PADDR2. See Section 14.3.5.13, “Physical Address Low (ETH_PADDR1) Register” and Section 14.3.5.14, “Physical Address High (ETH_PADDR2) Register.”
- ETH_OP_PAUSE (only needed for FDX flow control). See Section 14.3.5.15, “Opcode/Pause Duration (ETH_OP_PAUSE) Register.”
- ETH_R_CNTRL. See Section 14.3.5.10, “Receive Control (ETH_R_CNTRL) Register.”
- ETH_X_CNTRL. See Section 14.3.5.12, “Transmit Control (ETH_X_CNTRL) Register.”
- ETH_MII_SPEED (optional). See Section 14.3.5.8, “MII Speed Control (ETH_MII_SPEED) Register.”
- Clear MIB_RAM (locations 0x0200–0x02E3).

FEC FIFO/DMA registers requiring initialization are as follows:

- Initialize ETH_R_FSTART (optional). See Section 14.3.5.22, “FIFO Receive Start (ETH_R_FSTART) Register.”
- Initialize ETH_R_BUFF_SIZE. See Section 14.3.5.25, “Receive Buffer Size (ETH_R_BUFF_SIZE) Register.”
- Initialize ETH_R_DES_START. See Section 14.3.5.23, “Beginning of Receive Descriptor Ring (ETH_R_DES_START) Register.”
- Initialize ETH_X_DES_START. See Section 14.3.5.24, “Beginning of Transmit Descriptor Ring (ETH_X_DES_START) Register.”
- Initialize ETH_DMA_CONTROL. See Section 14.3.5.26, “DMA Function Control (ETH_DMA_CONTROL) Register.”
- Initialize (empty) transmit descriptor ring
- Initialize (empty) receive descriptor ring

14.4.1.1 Descriptor Controller Initialization

In the fast Ethernet controller, the descriptor control RISC initializes some registers after ETHER_EN is asserted. After the descriptor controller initialization sequence is complete, the hardware is ready for operation.

The following describes the FEC RISC initialization operations.

1. Initialize Backoff random number seed.
2. Activate receiver.
3. Activate transmit.

The following describes the FEC RISC initialization operations specific to the FEC.

1. Clear transmit FIFO.
   X_LAG, X_READ, X_WRITE = 0
2. Clear receive FIFO.
   R_LAG, R_READ, R_WRITE = ETH_R_FSTART
3. Initialize receive ring pointer.
   RDES_ADDR = ETH_R_DES_START
4. Initialize transmit ring pointer.
   XDES_ADDR = ETH_X_DES_START
5. Initialize FIFO count registers.
   R_COUNT = X_COUNT = 0

14.4.1.2 Initialization (After Asserting ETHER_EN)

After asserting the ETHER_EN bit of the ETH_ECNTRL register, software can set up the buffer/frame descriptors and write to the ETH_X_DES_ACTIVE and ETH_R_DES_ACTIVE registers.

14.5 Buffer Descriptors

14.5.1 Driver/DMA Operation with Buffer Descriptors

The data for the FEC frames must reside in memory external to the FEC. The data for a frame is placed in one or more buffers. A buffer descriptor (BD) that contains a starting address (pointer) and data length for the buffer is associated with each buffer. In addition to pointing to the buffer, the most significant bit of the BD is an ownership bit, which defines the current state of the buffer. Other bits in the buffer descriptor are used to communicate status/control information between the Ethernet MAC and the driver. To permit maximum user flexibility, the BDs used by the FEC DMA engineers also located in external memory.

Software produces buffers by allocating/initializing memory and initializing buffer descriptors. Setting the R/E (ownership) bit in the most significant word of the transmit (receive) buffer descriptor produces the buffer. A software write to either the ETH_X_DES_ACTIVE or the ETH_R_DES_ACTIVE register tells the FEC that a buffer has been placed in external memory for the transmit or receive data traffic, respectively. The hardware reads the BDs and consumes the buffers after they have been produced. After the data DMA is complete and the buffer descriptor status bits have been written by the DMA engine, the R(E) bit is cleared by hardware to signal the buffer has been consumed. Software may poll the BDs to detect when the buffers have been consumed or may rely on the buffer/frame interrupts. These buffers may then be processed by the driver and returned to the free list.

The ETHER_EN signal operates as a reset to the BD/DMA logic. When ETHER_EN is deasserted, the DMA engine BD pointers are reset to point to the starting transmit and receive BDs. The buffer descriptors are not initialized by hardware during reset. At least one transmit and receive buffer descriptor must be initialized by software (write 0x0000_0000 to the most significant word of buffer descriptor) before the ETHER_EN bit is set.

The buffer descriptors operate as a ring. The ETH_R_DES_START register defines the starting address for receive BDs. The ETH_X_DES_START register defines the starting address for transmit BDs. The last buffer descriptor in each ring is defined by the wrap (W) bit. When set, w indicates that the next descriptor in the ring is at the location pointed to by ETH_R_DES_START and ETH_X_DES_START for the receive and transmit rings, respectively. Buffer descriptor rings must start on a 32-bit boundary; it is strongly recommended that they are made 128-bit aligned.

14.5.1.1 Driver/DMA Operation with Transmit BDs

Typically, a transmit frame is divided between multiple buffers. An example is to have an Ethernet/802.3 header in the first buffer, an IP header in a second buffer, a TCP header in a third buffer, and an application payload in the last buffer. The Ethernet MAC does not prepend the Ethernet header (destination address, source address, length/type fields), so this must be provided by the driver in one of the transmit buffers. The Ethernet MAC can append the Ethernet CRC to the frame. Whether the CRC is appended by the MAC or by the driver is determined by the TC bit in the transmit BD, which must be set by the driver. When the DMA of the transmit frame is complete, the DMA controller appends a control word to the frame. The requirement for the control word is that the TC and ABC bits must be in the same position, as defined by the transmit BD. The simplest solution is to copy the most significant 32 bits of the transmit BD into the transmit FIFO at the end of the frame.

In a typical end station application, the TC bit always equals 1. For a switch/router application, the TC bit may be 1 or 0, depending on what type of port the frame arrived on and whether the frame contents were modified. The append bad CRC (ABC) bit is 0 unless an error has occurred (for example, a data parity error during DMA transfer) that results in data corruption.

The driver (TxBD software producer) should set up TxBDs in such a way that a complete transmit frame is given to the hardware at once. If a transmit frame consists of three buffers, the BDs should be initialized with pointer, length, and control (W, L, TC, ABC), and then the ownership (R) bits should be set equal to 1 in reverse order (BD 3, BD 2, BD 1) to ensure the complete frame is ready in memory before the DMA begins. If the TxBDs are set up in order, the DMA Controller could DMA the first BD before the second was made available, potentially causing a transmit FIFO underrun.

In the FEC, the driver notifies the DMA that new transmit frame(s) are available by writing to the ETH_X_DES_ACTIVE register. When data is written to this register (data value is not significant), the FEC RISC tells the DMA to read the next transmit BD in the ring. After the start, the RISC + DMA continue reading and interpreting transmit BDs in order and DMA the associated buffers until a transmit BD is encountered with the R bit cleared to 0. At this point, the FEC polls this BD one more time. If the R bit equals 0 a second time, the RISC stops the transmit descriptor read process until software sets up another transmit frame and writes to ETH_X_DES_ACTIVE.

When the DMA of each transmit buffer is complete, the DMA writes back to the BD to clear the R (ownership) bit, indicating that the hardware consumer is finished with the buffer. A second driver task (TxBD software consumer) processes the transmit descriptor ring and return buffers consumed by the hardware to the free list.

14.5.1.2 Driver/DMA Operation with Receive BDs

Unlike transmit, the length of the receive frame is unknown by the driver ahead of time. Therefore, the driver must set a variable to define the length of all receive buffers. In the FEC, this variable is written to the ETH_R_BUFF_SIZE register.

The driver (receive BD software producer) should set up some number of empty buffers for the Ethernet by initializing the address field and the E and W bits of the associated receive BDs. The hardware (receive DMA) consumes these buffers by filling them with data as frames are received and clearing the E bit and writing to the Lbit (1 indicates last buffer in frame), the frame status bits (if L= 1), and the length field.

If a receive frame spans multiple receive buffers, the L bit is only set for the last buffer in the frame. For any other buffer, the length field in the receive BD is written with the default receive buffer length value by the DMA (at the same time the e bit is cleared). For end-of-frame buffers, the receive BD is written with L set and information written to the status bits (M, BC, MC, LG, NO, SH, CR, OV.TR). Some of the status bits are error indicators that, if set, indicate the receive frame should be discarded and not given to higher layers. The frame status/length information is written into the receive FIFO following the end of the frame (as a single 32-bit word) by the receive logic. The length field for the end-of-frame buffer is written with the length of the entire frame, not the length of the last buffer.

For simplicity, the driver may assign the default receive buffer length to be large enough to contain an entire frame, keeping in mind that a malfunction on the network or out-of-specification implementation could result in giant frames. Frames of 2 KB (2048 bytes) or larger are truncated by the FEC at 2047 bytes, so software is guaranteed never to see a receive frame larger than 2047 bytes.

Similar to transmit, the FEC polls the receive descriptor ring after the driver sets up receive BDs and writes to the ETH_R_DES_ACTIVE register. As frames are received, the FEC fills receive buffers and updates the associated BDs, and then reads the next BD in the receive descriptor ring. If the FEC reads a receive BD and finds the E bit equal to 0, it polls this BD once more. If the BD equals 0 a second time, the FEC stops reading receive BDs until the driver writes to ETH_R_DES_ACTIVE.

14.5.2 Ethernet Receive Buffer Descriptor (RxBD)

In the RxBD, initialize the E and W bits in the first word and the pointer in the second word. When the buffer has been filled by DMA, the Ethernet controller modifies the E, L, M, BC, MC, LG, NO, CR, OV, and TR bits and also writes the data length into the first word. If a single receive buffer descriptor is used, the data length is the portion of the data buffer used (the total length of the frame). If the received message spans several data buffers, the data length in all but the last receive buffer (those with the L bit equaling 0) is ETH_R_BUFF_SIZE. The data length of the last receive buffer descriptor (the descriptor where the L bit is equal to 1) is the total length of the frame. The M, BC, MC, LG, NO, CR, OV, and TR bits in the first word of the buffer descriptor are only modified by the Ethernet controller when the l bit is set.

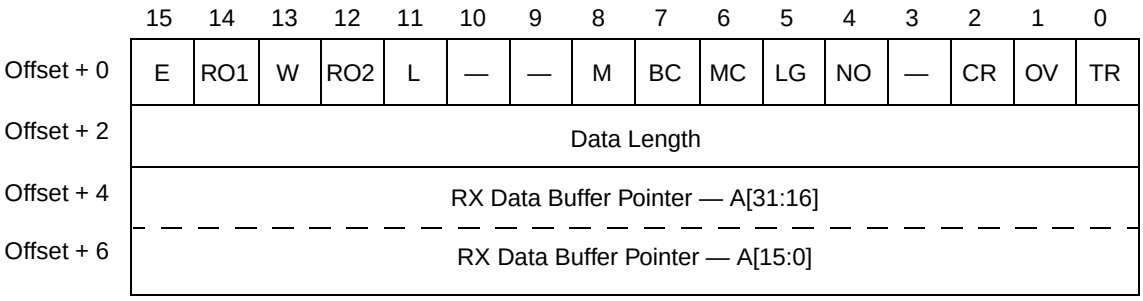


Figure 14-28. RxBD — Receive Buffer Descriptor

The first word of the RXBD contains control and status bits. Its format is detailed in Table 14-32.

Table 14-32. RxBD — Receive Buffer Field Descriptions

Field	Description
E	Empty, written by the FEC (=0) and user (=1). 0 The data buffer associated with this BD has been filled with received data or data reception has been aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty or reception is currently in progress.
RO1	Receive software ownership bit This field is reserved for use by software. This read/write bit must not be modified by hardware. Its value does not affect hardware.
W	Wrap, written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in R_DES_START.
RO2	Receive software ownership bit This field is reserved for use by software. This read/write bit must not be modified by hardware. Its value does not affect hardware.
L	Last in frame, written by the FEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
M	Miss, written by the FEC. This bit is set by the FEC for frames that were accepted in promiscuous mode, but were flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, you can use the M bit to determine whether the frame was destined to this station. This bit is valid only if the L and the PROM bits are set. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.
BC	Set if the DA is broadcast (FF-FF-FF-FF-FF-FF).
MC	Set if the DA is multicast and not broadcast.
LG	RX frame length violation, written by the FEC. A frame length greater than ETH_R_CNTRL[MAX_FL] was recognized. This bit is valid only if the L bit is set. The receive data is not altered in any way unless the length exceeds 2047 bytes.
NO	RX non-octet aligned frame, written by the FEC. A frame that contains a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error. This bit is valid only if the L bit is set. If this bit is set, the CR bit is not set.
CR	RX CRC error, written by the FEC. This frame contains a CRC error and is an integral number of octets in length. This bit is valid only if the l bit is set.

Table 14-32. RxBD — Receive Buffer Field Descriptions (Continued)

Field	Description
OV	Overrun, written by the FEC. A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits (M, LG, NO, SH, CR, and CL) lose their normal meaning and is zero. This bit is valid only if the L bit is set.
TR	Set if the receive frame is truncated (frame length > 2047 bytes). If the TR bit is set, the frame should be discarded and the other error bits should be ignored because they may be incorrect.
Data Length	Written by the FEC. If a single receive buffer descriptor is being used, the data length is the number of octets written by the FEC into this BD's data buffer (the total length of the frame). If the received message spans several data buffers, the data length is ETH_R_BUFF_SIZE in all but the last receive buffer descriptor (those with the L bit equaling 0). The data length of the last receive buffer descriptor (the descriptor where the L bit is equal to 1) is the total length of the frame in octets.
Rx Buffer Pointer	Written by user. The receive buffer pointer, which always points to the first location of the associated data buffer, must always be evenly divisible by 16. The buffer must reside in memory external to the FEC.

Note: Any time the software driver sets an E bit in one or more receive descriptors, the driver should follow that with a write to ETH_R_DES_ACTIVE.

14.5.3 Ethernet Transmit Buffer Descriptor

Data is presented to the FEC for transmission by arranging the data in buffers referenced by the channel's TxBDS. The Ethernet controller confirms transmission by clearing an ownership bit (R bit) when the buffer's DMA is complete. In the TxBD, initialize the R, W, L, and TC bits and the length (in bytes) in the first word and the buffer pointer in the second word.

The FEC clears the R bit equal to 0 in the first word of the BD when the buffer has been addressed by the direct memory access controllers. Status bits for the buffer/frame are not included in the transmit buffer descriptors. Transmit frame status is indicated via individual interrupt bits (error conditions) and in statistic counters in the MIB block.

The TxBD fields are detailed in Figure 14-29.

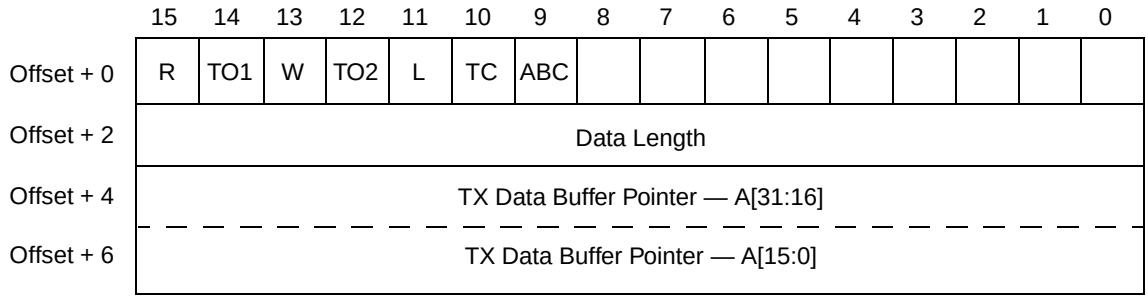


Figure 14-29. Transmit Buffer Descriptor (TxBD)

NOTE

The TX data buffer pointer must be evenly divisible by 4.

Table 14-33. Transmit Buffer Field Descriptions

Field	Description
R	Ready, written by FEC and user. 0 The data buffer associated with this BD is not ready for transmission. You are free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer that was prepared for transmission has not been transmitted, or is currently being transmitted. No fields of this BD may be modified after this bit is set.
TO1	Transmit software ownership bit. This field is reserved for use by software. This read/write bit must not be modified by hardware. Its value does not affect hardware.
W	Wrap, written by user. This field is reserved for use by software. This read/write bit must not be modified by hardware. Its value does not affect hardware. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in ETH_X_DES_START.
TO2	Transmit software ownership bit This field is reserved for use by software. This read/write bit must not be modified by hardware. Its value does not affect hardware.
L	Last in frame, written by user. 0 The buffer is not the last in the transmit frame. 1 The buffer is the last in the transmit frame.
TC	TX CRC, written by user (only valid if L equals 1). 0 End transmission immediately after the last data byte. 1 Transmit the CRC sequence after the last data byte.
ABC	Append bad CRC, written by user (only valid if l = 1) 0 No effect. 1 Transmit an invalid CRC sequence after the last data byte (regardless of TC value).
Data Length	Data length, written by user. Data length is the number of octets the FEC should transmit from this BD's data buffer. It is never modified by the FEC. Bits [21:31] are used by the DMA engine. Bits [16:20] are ignored.
Transmit Date Buffer Pointer	TX buffer pointer, written by user. The transmit buffer pointer, which contains the address of the associated data buffer, must always be evenly divisible by 4. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

Note: After the software driver has set up the buffers for a frame, it should set up the corresponding BDs. The last step in setting up the BDs for a transmit frame should be to set the r bit in the first BD for the frame. The driver should follow that with a write to ETH_X_DES_ACTIVE, which triggers the FEC to poll the next BD in the ring.

14.6 Network Interface Options

Each FEC supports an MII interface and reduced MII interface for 10/100 Mbit/s Ethernet and a 7-wire serial interface for 10 Mbit/s Ethernet. The interface mode is selected by the MII_MODE and RMII_MODE bit in the ETH_R_CNTRL register. In MII mode, (ETH_R_CNTRL[MII_MODE] = 1, ETH_R_CNTRL[RMII_MODE] = 0), there are 18 signals defined by the 802.3 standard and supported by the EMAC. Table 14-34 shows these.

Table 14-34. MII Interface

Signal Description	FEC signal name
Transmit clock	FEC*n*_TX_CLK
Transmit enable	FEC*n*_TX_EN
Transmit data	FEC*n*_TXD[3:0]
Transmit error	FEC*n*_TX_ER
Collision	FEC*n*_COL
Carrier sense	FEC*n*_CRS
Receive clock	FEC*n*_RX_CLK
Receive enable	FEC*n*_RX_DV/RMII_CRS_DV
Receive data	FEC*n*_RXD[3:0]
Receive error	FEC*n*_RX_ER/RMII_RX_ER
Management channel clock	FEC*n*_MDC
Management channel serial data	FEC*n*_MDIO

In RMII mode, (ETH_R_CNTRL[MII_MODE] = 1, ETH_R_CNTRL[RMII_MODE] = 1), the FEC supports 10 external signals. These signals are shown in Table 14-35.

Table 14-35. RMII Interface

Signal Description	FEC signal name
Reference Clock	FEC*n*_TX_CLK/RMII_REF_CLK
Transmit enable	FEC*n*_TX_EN/RMII_TX_EN
Transmit data	FEC*n*_TXD[1:0]/RMII_TX[1:0]
Receive data valid/ Carrier Sense	FEC*n*_RX_DV/RMII_CRS_DV
Receive data	FEC*n*_RXD[1:0]/RMII_RX[1:0]
Receive error	FEC*n*_RX_ER/RMII_EX_ER
Management channel clock	FEC*n*_MDC
Management channel serial data	FEC*n*_MDIO

Serial mode connections to the external transceiver are shown in Table 14-36.

Table 14-36. 7-Wire Interface

Signal Description	FEC signal name
Transmit clock	FEC*n*_TX_CLK/RMII_REF_CLK
Transmit enable	FEC*n*_TX_EN/RMII_TX_EN
Transmit data	FEC*n*_TXD_0/RMII_RX0
Collision	FEC*n*_COL
Receive clock	FEC*n*_RX_CLK

Table 14-36. 7-Wire Interface (Continued)

Signal Description	FEC signal name
Receive enable	FEC*n*_RX_DV/RMII_CRS_DV
Receive data	FEC*n*_RXD_0/RMII_RX0

14.6.1 FEC Frame Transmission

The Ethernet transmitter is designed to work with almost no intervention from software. After ETHER_EN is asserted and data appears in the transmit FIFO, the Ethernet MAC is able to transmit onto the network.

When the transmit FIFO fills to the watermark (defined by the ETH_X_WMRK register), the MAC transmit logic asserts TX_EN and starts transmitting the preamble sequence, the start frame delimiter, and then the frame information from the FIFO. However, the controller defers the transmission if the network is busy (carrier sense is asserted). Before transmitting, the controller waits for carrier sense to become inactive, and then determines if carrier sense stays inactive for 60 bit times. If so, then the transmission begins after waiting an additional 36 bit times (96 bit times after carrier sense originally became inactive).

If a collision occurs during transmission of the frame (half-duplex mode), the Ethernet controller follows the specified backoff procedures and attempts to retransmit the frame until the retry limit threshold is reached. The transmit FIFO stores at least the first 64 bytes of the transmit frame, so the first 64 bytes do not have to be retrieved again from system memory in case of a collision. This improves bus utilization and latency in case immediate retransmission is necessary.

When all the frame data has been transmitted, the FCS (32-bit CRC) bytes are appended if the TC bit is set in the transmit frame control word. If the ABC bit is set in the transmit frame control word, a bad CRC is appended to the frame data regardless of the TC bit value. Following the transmission of the CRC, the Ethernet controller writes the frame status information to the MIB block. Short frames are automatically padded by the transmit logic (if the TC bit in the transmit buffer descriptor for the end-of-frame buffer equals 1).

Both buffer (TXB, FEC only) and frame (TFINT, FEC) interrupts may be generated as determined by the settings in the ETH_IMASK register.

Transmit error interrupts are HBERR, BABT, LATE_COL, COL_RETRY_LIM, XFIFO_UN and XFIFO_ERROR. If the transmit frame length exceeds MAX_FL bytes, the BABT interrupt is asserted; however, the entire frame is transmitted (no truncation).

To pause transmission, set the graceful transmit stop (GTS) bit in the ETH_X_CNTRL register. When the GTS is set, the FEC transmitter stops immediately if transmission is not in progress; otherwise, it continues transmission until the current frame finishes or terminates with a collision. After the transmitter has stopped, the GRA interrupt is asserted. If GTS is cleared, the FEC resumes transmission with the next frame.

The Ethernet controller transmits bytes least significant bit (LSB) first.

14.6.1.1 Duplicate Frame Transmission

The FEC fetches transmit buffer descriptors (TxBDs) and the corresponding transmit data continuously until the transmit FIFO is full. It does not determine whether the TxBD to be fetched is already being processed internally (as a result of a wrap). As the FEC nears the end of the transmission of one frame, it begins to DMA the data for the next frame. To remain one BD ahead of the DMA, it also fetches the TxBD for the next frame. It is possible that the FEC fetches from memory a BD that has already been processed but not yet written back (it is read a second time with the R bit remains set). In this case, the data is fetched and transmitted again.

Using at least three TxBDs fixes this problem for large frames, but not for small frames. To ensure correct operation for large or small frames, one of the following must be true:

- The FEC software driver ensures that there is always at least one TxBD with the ready bit cleared.
- Every frame uses more than one TxBD and every TxBD but the last is written back immediately after the data is fetched.
- The FEC software driver ensures a minimum frame size, *n*. The minimum number of TxBDs is then (Tx FIFO Size ÷ (*n* + 4)) rounded up to the nearest integer (though the result cannot be less than three). The default Tx FIFO size is 192 bytes; this size is programmable.

14.6.2 FEC Frame Reception

The FEC receiver is designed to work with almost no intervention from the host and can perform address recognition, CRC checking, short frame checking, and maximum frame length checking.

When the driver enables the FEC receiver by asserting ETHER_EN, it immediately starts processing receive frames. When RX_DV asserts, the receiver checks for a valid PA/SFD header. If the PA/SFD is valid, it is stripped and the frame is processed by the receiver. If a valid PA/SFD is not found, the frame is ignored.

NOTE

The FEC receive block transfers blocks of 16 bytes to the receive buffer even if the message length is not divisible by 16 bytes. Therefore, if the message length is not divisible by 16 bytes, extra bytes are added.

In serial mode, the first 16 bit times of RX_D0 following assertion of RX_DV (RENA) are ignored. Following the first 16 bit times, the data sequence is checked for alternating 1/0s. If a 11 or 00 data sequence is detected during bit times 17 to 21, the remainder of the frame is ignored. After bit time 21, the data sequence is monitored for a valid SFD (11). If a 00 is detected, the frame is rejected. When a 11 is detected, the PA/SFD sequence is complete.

In MII mode, the receiver checks for at least one byte matching the SFD. Zero or more PA bytes may occur, but if a 00 bit sequence is detected prior to the SFD byte, the frame is ignored.

After the first six bytes of the frame have been received, the FEC performs address recognition on the frame.

After a collision window (64 bytes) of data has been received, and if address recognition has not rejected the frame, the receive FIFO is signalled that the frame has been accepted and may be passed on to the

DMA. If the frame is a runt (due to collision) or is rejected by address recognition, the receive FIFO is notified to reject the frame. Thus, no collision fragments are presented except late collisions, which indicate serious LAN problems.

During reception, the Ethernet controller checks for various error conditions and after the entire frame is written into the FIFO, a 32-bit frame status word is written into the FIFO. This status word contains the M, BC, MC, LG, NO, SH, CR, OV and TR status bits and the frame length.

Receive buffer (RXB, FEC only) and frame (RFINT, FEC only) interrupts may be generated if enabled by the ETH_IMASK register. BABR and RFIFO_ERROR are receive error interrupts. Receive frames are not truncated if they exceed the MAX_FL byte length; however, the BABR interrupt occurs and the LG bit in the receive BD is set.

When the receive frame is complete, the FEC sets the l bit in the receive BD, writes the other frame status bits into the receive BD, and clears the E bit. Next, the Ethernet controller generates a maskable interrupt (RFINT bit in ETH_IEVENT, maskable by RFIEN bit in ETH_IMASK), indicating a frame has been received and is in memory. The Ethernet controller then waits for a new frame.

The Ethernet controller receives serial data LSB first.

14.6.3 Ethernet Address Recognition

The FEC filters the received frames based on destination address (DA) type — individual (unicast), group (multicast), or broadcast (all-ones group address). The difference between an individual address and a group address is determined by the I/G bit in the destination address field. A flowchart for address recognition on received frames is illustrated in the following figures.

Address recognition is accomplished through the use of the receive block and microcode running on the descriptor controller. The flowchart shown in Figure 14-30 illustrates the address recognition decisions made by the receive block, while Figure 14-31 illustrates the decisions made by the descriptor controller.

If the DA is a broadcast address and broadcast reject (BC_REJ bit is deasserted, the frame is accepted unconditionally, as shown in Figure 14-30. Otherwise, if the DA is not a broadcast address, the descriptor controller runs the address recognition subroutine, as shown in Figure 14-31.

If the DA is a group (multicast) address and flow control is disabled, the descriptor controller performs a group hash table lookup using the 64-entry hash table programmed in ETH_GADDR1 and ETH_GADDR2. If a hash match occurs, address recognition hash match bar (AR_HM_B) is set to 0 and the receiver accepts the frame. If flow control is enabled, the descriptor controller does an exact address match check between the DA and the designated pause DA in registers FDXFC_DA1 and FDXFC_DA2. If a pause DA exact match occurs, the address recognition exact match bar (AR_EM_B) is set to 0. If the receive block determines the received frame is a valid pause frame, the frame is rejected. The receiver detects a pause frame with the DA field set to the designated pause DA or the unicast physical address.

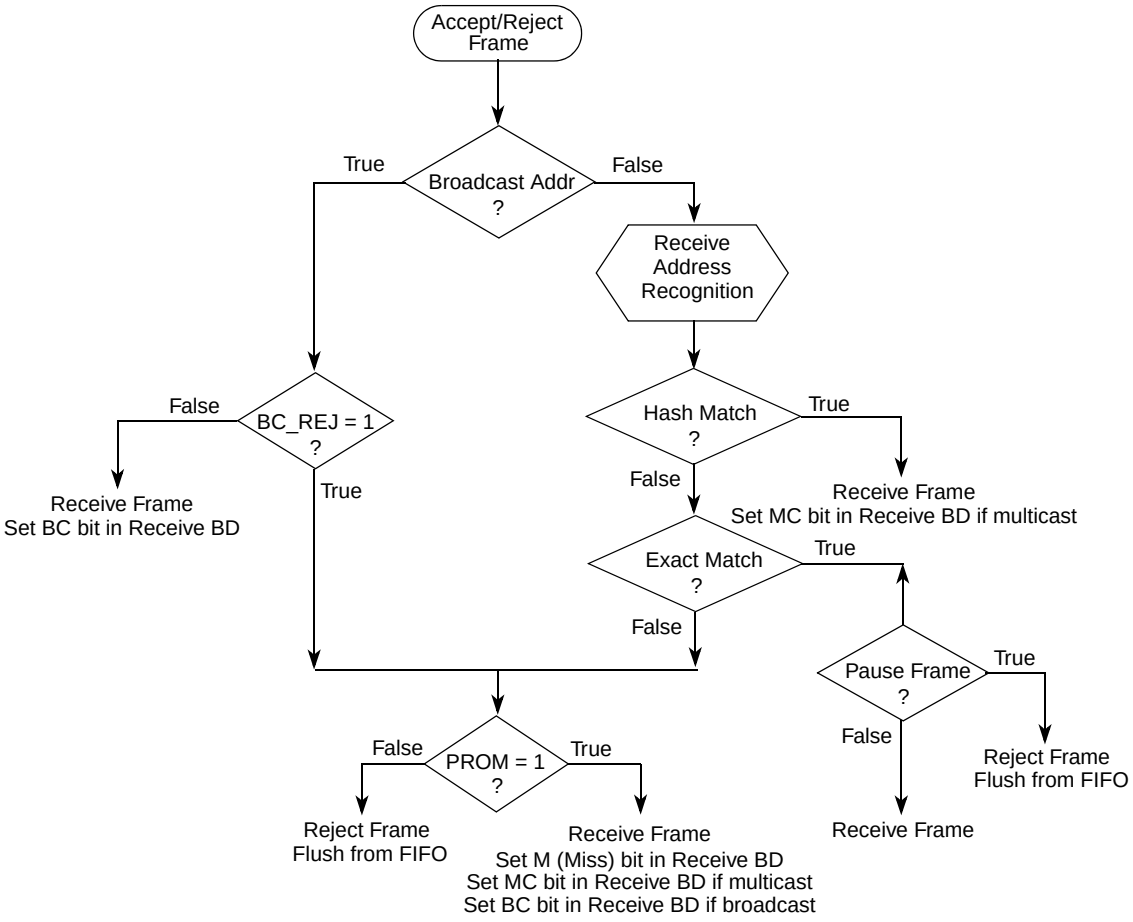
If the DA is the individual (unicast) address, the descriptor controller performs an individual exact match comparison between the DA and 48-bit physical address that you program in the ETH_PADDR1 and ETH_PADDR2 registers. If an exact match occurs, AR_EM_B is set to 0; otherwise, the descriptor controller does an individual hash table lookup using the 64-entry hash table programmed in the

ETH_IADDR1 and ETH_IADDR2 registers. In the case of an individual hash match, AR_HM_B is set to 0. Again, the receiver accepts or rejects the frame based on pause frame detection, shown in Figure 14-30.

If neither a hash match (group or individual), nor an exact match (group or individual) occur, then both ar_hm_b and ar_em_b are set to 1. In this case, if promiscuous mode is enabled (r_cntrl.prom = 1), the frame is accepted and the MISS bit in the receive buffer descriptor is set. Otherwise, the frame is rejected and the MISS bit is cleared.

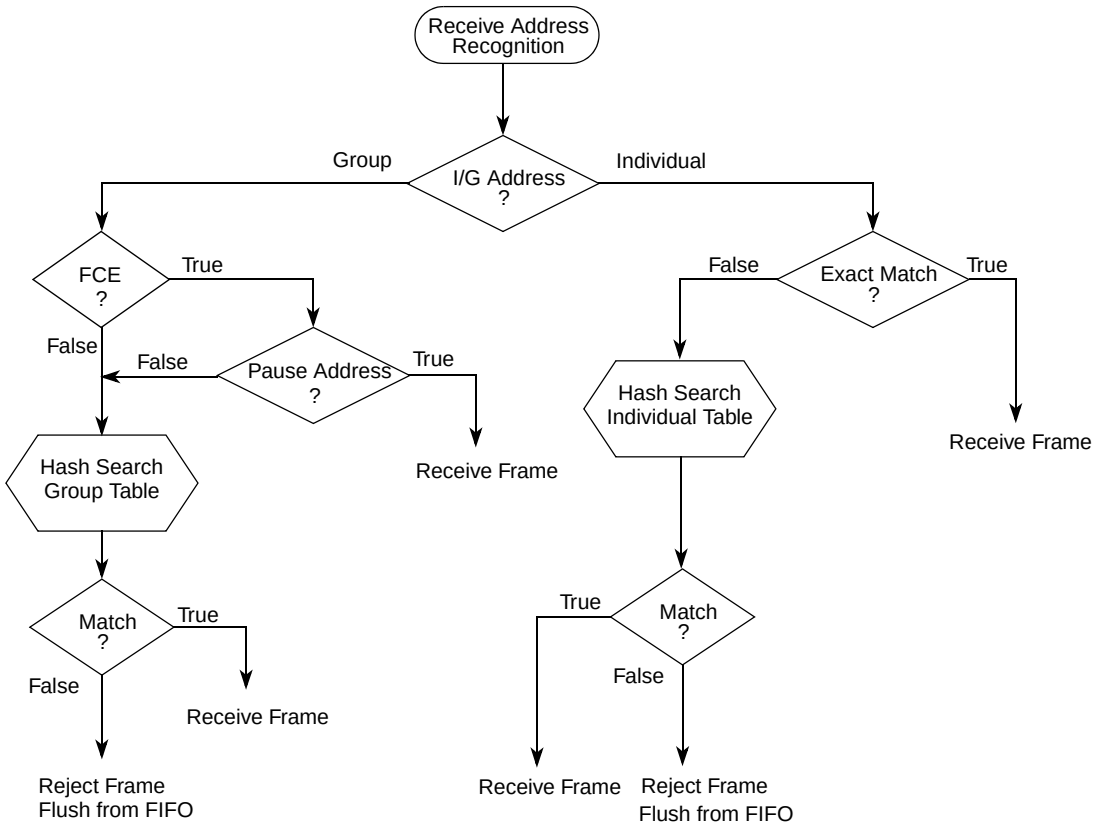
Similarly, if the DA is a broadcast address, broadcast reject (ETH_R_CNTRL[BC_REJ]) is asserted, and promiscuous mode is enabled, the frame is accepted, and the MISS bit in the receive buffer descriptor is set. Otherwise, the frame is rejected and the MISS bit is cleared.

In general, when a frame is rejected, it is flushed from the FIFO.



NOTES:
BC_REJ — field in ETH_R_CNTRL register (BroadCast REJect)
PROM — field in ETH_R_CNTRL register (PROMiscous mode)
Pause Frame — valid PAUSE frame received

Figure 14-30. Ethernet Address Recognition — Receive Block Decisions



NOTES:
FCE — field in ETH_R_CNTRL register (Flow Control Enable)
I/G — Individual/Group bit in Destination Address (least significant bit in first byte received in MAC frame)

Figure 14-31. Ethernet Address Recognition — Microcode Decisions

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits, which are represented by 64 bits stored in ETH_GADDR1/ETH_GADDR2 (group address hash match) or ETH_IADDR1/ETH_IADDR2 (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the 6 most significant bits of the CRC-encoded result to generate a number between 0 and 63. The MSB of the CRC result selects ETH_GADDR1 (MSB = 1) or ETH_GADDR2 (MSB = 0). The least significant 5 bits of the hash result select the bit within the selected register. If the CRC generator selects a bit set in the hash table, the frame is accepted. Otherwise, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (or 87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The hash table registers must be initialized. You must compute the hash for a particular address in

software, retrieve the result and use it to program the FEC hash table registers. The CRC32 polynomial to use in computing the hash is:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^4 + X^2 + X + 1 \qquad \textit{Eqn. 14-1}$$

Table 14-37 includes example destination addresses and corresponding hash values for reference.

Table 14-37. Destination Address to 6-Bit Hash

48-Bit DA	6-Bit Hash (in Hex)	Hash Decimal Value
65:ff:ff:ff:ff:ff	0x00	0
55:ff:ff:ff:ff:ff	0x01	1
15:ff:ff:ff:ff:ff	0x02	2
35:ff:ff:ff:ff:ff	0x03	3
b5:ff:ff:ff:ff:ff	0x04	4
95:ff:ff:ff:ff:ff	0x05	5
d5:ff:ff:ff:ff:ff	0x06	6
f5:ff:ff:ff:ff:ff	0x07	7
db:ff:ff:ff:ff:ff	0x08	8
fb:ff:ff:ff:ff:ff	0x09	9
bb:ff:ff:ff:ff:ff	0xoA	10
8b:ff:ff:ff:ff:ff	0x0B	11
0b:ff:ff:ff:ff:ff	0x0C	12
3b:ff:ff:ff:ff:ff	0x0D	13
7b:ff:ff:ff:ff:ff	0x0E	14
5b:ff:ff:ff:ff:ff	0x0F	15
27:ff:ff:ff:ff:ff	0x10	16
07:ff:ff:ff:ff:ff	0x11	17
57:ff:ff:ff:ff:ff	0x12	18
77:ff:ff:ff:ff:ff	0x13	19
f7:ff:ff:ff:ff:ff	0x14	20
c7:ff:ff:ff:ff:ff	0x15	21
97:ff:ff:ff:ff:ff	0x16	22
a7:ff:ff:ff:ff:ff	0x17	23
99:ff:ff:ff:ff:ff	0x18	24
b9:ff:ff:ff:ff:ff	0x19	25
f9:ff:ff:ff:ff:ff	0x1A	26
c9:ff:ff:ff:ff:ff	0x1B	27
59:ff:ff:ff:ff:ff	0x1C	28
79:ff:ff:ff:ff:ff	0x1D	29

Table 14-37. Destination Address to 6-Bit Hash (Continued)

48-Bit DA	6-Bit Hash (in Hex)	Hash Decimal Value
29:ff:ff:ff:ff:ff	0x1E	30
19:ff:ff:ff:ff:ff	0x1F	31
d1:ff:ff:ff:ff:ff	0x20	32
f1:ff:ff:ff:ff:ff	0x21	33
b1:ff:ff:ff:ff:ff	0x22	34
91:ff:ff:ff:ff:ff	0x23	35
11:ff:ff:ff:ff:ff	0x24	36
31:ff:ff:ff:ff:ff	0x25	37
71:ff:ff:ff:ff:ff	0x26	38
51:ff:ff:ff:ff:ff	0x27	39
7f:ff:ff:ff:ff:ff	0x28	40
4f:ff:ff:ff:ff:ff	0x29	41
1f:ff:ff:ff:ff:ff	0x2A	42
3f:ff:ff:ff:ff:ff	0x2B	43
bf:ff:ff:ff:ff:ff	0x2C	44
9f:ff:ff:ff:ff:ff	0x2D	45
df:ff:ff:ff:ff:ff	0x2E	46
ef:ff:ff:ff:ff:ff	0x2F	47
93:ff:ff:ff:ff:ff	0x30	48
b3:ff:ff:ff:ff:ff	0x31	49
f3:ff:ff:ff:ff:ff	0x32	50
d3:ff:ff:ff:ff:ff	0x33	51
53:ff:ff:ff:ff:ff	0x34	52
73:ff:ff:ff:ff:ff	0x35	53
23:ff:ff:ff:ff:ff	0x36	54
13:ff:ff:ff:ff:ff	0x37	55
3d:ff:ff:ff:ff:ff	0x38	56
0d:ff:ff:ff:ff:ff	0x39	57
5d:ff:ff:ff:ff:ff	0x3A	58
7d:ff:ff:ff:ff:ff	0x3B	59
fd:ff:ff:ff:ff:ff	0x3C	60
dd:ff:ff:ff:ff:ff	0x3D	61
9d:ff:ff:ff:ff:ff	0x3E	62
bd:ff:ff:ff:ff:ff	0x3F	63

14.6.4 Full-Duplex Flow Control

Full-duplex flow control allows you to transmit pause frames and to detect received pause frames. Upon detection of a pause frame, MAC data frame transmission stops for a given pause duration.

To enable pause frame detection, the FEC must operate in full-duplex mode (ETH_X_CNTRL[FDEN] set) and flow control enable (ETH_R_CNTRL[FCE]) must be set. The FEC detects a pause frame when the fields of the incoming frame match the pause frame specifications, as shown in Table 14-38. In addition, the receive status associated with the frame should indicate that the frame is valid.

Table 14-38. PAUSE Frame Field Specification

48-Bit Destination Address	0x0180_C200_0001 or Physical Address
48-bit source address	any
16-bit type	0x8808
16-bit opcode	0x0001
16-bit pause duration	0x0000 to 0xFFFF

Pause frame detection is performed by the receiver and descriptor controller modules. The descriptor controller runs an address recognition subroutine to detect the specified pause frame destination address, while the receiver detects the type and opcode pause frame fields. On detection of a pause frame, graceful transmit stop is asserted by the FEC internally. When transmission has paused, the graceful stop complete (GRA) interrupt is asserted and the pause timer begins to increment. The pause timer makes use of the transmit backoff timer hardware, which is used for tracking the appropriate collision backoff time in half-duplex mode. The pause timer increments once every slot time, until pause_duration slot times have expired. When pause_duration expires, graceful transmit stop is deasserted, allowing MAC data frame transmission to resume. The receive flow control pause (RFC_PAUSE) status bit is asserted while the transmitter is paused due to reception of a pause frame.

To transmit a pause frame, the FEC must operate in full-duplex mode and software must assert flow control pause (TFC_PAUSE). On assertion of TFC_PAUSE, the transmitter asserts graceful transmit stop internally. When the transmission of data frames stops, the graceful stop complete (GRA) interrupt asserts. Following GRA assertion, the pause frame is transmitted. When pause frame transmission is complete, TFC_PAUSE and graceful transmit stop are deasserted internally.

During pause frame transmission, the transmit hardware places data into the transmit data stream from the registers shown in Table 14-39.

Table 14-39. Transmit Pause Frame Registers

PAUSE Frame Fields	FEC Register	Register Contents
48-bit destination address	{FDXFC_DA1[0:31], fDXFC_DA2[0:15]}	0x0180_C200_0001
48-bit source address	{ETH_PADDR1[0:31], ETH_PADDR2[0:15]}	Physical address
16-bit type	ETH_PADDR2[16:31]	8808
16-bit opcode	ETH_OP_PAUSE[0:15]	0x0001
16-bit pause duration	ETH_OP_PAUSE[16:31]	0x0000 to 0xFFFF

Specify the desired pause duration in the ETH_OP_PAUSE register.

When the transmitter is paused due to receiver/descriptor controller pause frame detection, transmit flow control pause (TFC_PAUSE) may continue to be asserted and cause the transmission of a single pause frame. In this case, the GRA interrupt is not asserted.

14.6.5 Inter-Packet Gap Time

The minimum inter-packet gap time for back-to-back transmission is 96 bit times. After completing a transmission or after the backoff algorithm completes, the transmitter waits for carrier sense to be negated before starting its 96 bit time IPG counter. Frame transmission may begin 96 bit times after carrier sense is negated if it stays negated for at least 60 bit times. If carrier sense asserts during the last 36 bit times, it is ignored and a collision occurs.

The receiver receives back-to-back frames with a minimum spacing of at least 28 bit times. If an inter-packet gap between receive frames is less than 28 bit counts, the following frame may be discarded by the receiver.

14.6.6 Collision Handling

If a collision occurs during frame transmission, the Ethernet controller continues the transmission for at least 32 bit times, transmitting a jam pattern consisting of 32 ones. If the collision occurs during the preamble sequence, the jam pattern is sent after the end of the preamble sequence.

If a collision occurs within 64 byte times, the retry process is initiated. The transmitter waits a random number of slot times. A slot time is 512 bit times. If a collision occurs after 64 byte times, no retransmission is performed and the end-of-frame buffer is closed with an LC error indication.

14.6.7 MII Internal and External Loopback

Internal and external loopback are supported by the Ethernet controller. In loopback mode, both of the FIFOs are used and the FEC actually operates in a full-duplex fashion. Internal and external loopback are configured using combinations of the LOOP and DRT bits in the ETH_R_CNTRL register and the FDEN bit in the ETH_X_CNTRL register.

For internal and external loopback, set FDEN equal to 1.

For internal loopback, set LOOP equal to 1 and DRT equal to 0. TX_EN and TX_ER do not assert during internal loopback. During internal loopback, the transmit/receive data rate is higher than in normal operation because the internal system clock is used by the transmit and receive blocks instead of the clocks from the external transceiver. This causes an increase in the required system bus bandwidth for transmit and receive data being direct memory addressed to/from external memory. It may be necessary to pace the frames on the transmit side and/or limit the size of the frames to prevent transmit FIFO underrun and receive FIFO overflow.

For external loopback, set LOOP equal to 0 and DRT equal 0, and then configure the external transceiver for loopback.

14.6.8 RMII Loopback

The FEC also supports RMII loopback. In this mode, transmit data is sent to the RMII receive logic and out the FEC*n*_TXD[1:0] pins. To enable RMII loopback mode, set the ETH_R_CNTRL[RMII_MODE], ETH_R_CNTRL[MII_MODE], ETH_R_CNTRL[RMII_LOOP], and ETH_X_CNTRL[FDEN] bits; clear the ETH_R_CNTRL[RMII_ECHO] and ETH_R_CNTRL[LOOP] bits.

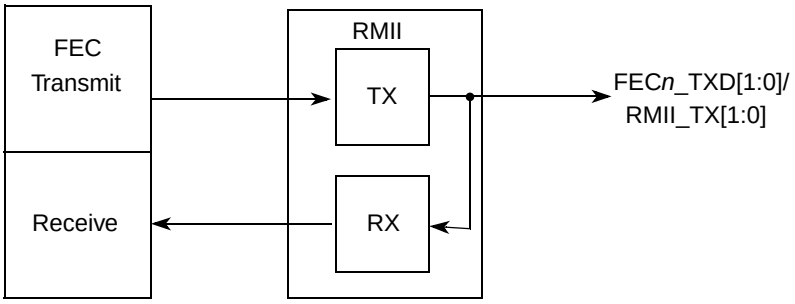


Figure 14-32. RMII Loopback Mode

14.6.9 RMII Echo

The FEC also supports RMII echo mode, which transmits data on FEC*n*_TXD[1:0]/RMII_TX[1:0] as it is received on FEC*n*_RXD[1:0]/RMII_RX[1:0]. To enable RMII loopback mode, set the ETH_R_CNTRL[RMII_MODE,] ETH_R_CNTRL[MII_MODE], and ETH_R_CNTRL[RMII_ECHO] bits; clear the ETH_R_CNTRL[RMII_ECHO], ETH_R_CNTRL[LOOP], and ETH_R_CNTRL[RMII_LOOP] bits.

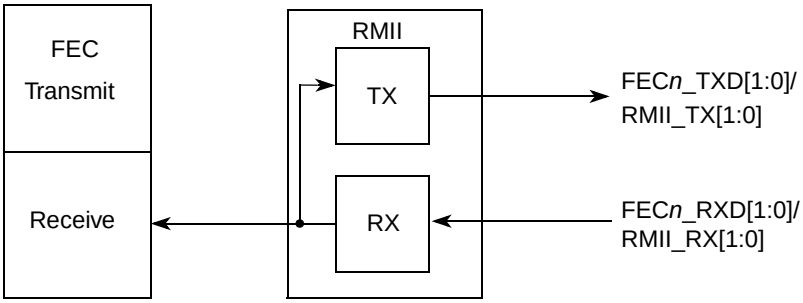


Figure 14-33. RMII Echo Mode

14.6.10 Ethernet Error-Handling Procedure

The Ethernet controller reports frame reception and transmission error conditions using the FEC receive BDs, the ETH_IEVENT register and the MIB block counters.

14.6.11 Transmission Errors

- Transmitter Underrun

- If this error occurs, the FEC sends 32 bits that ensure a CRC error and stops transmitting. All remaining buffers for that frame are then flushed and closed. The UN bit is set in the X_STATUS register. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame.
- The XFIFO_UN interrupt is asserted if enabled in the ETH_IMASK register

- Carrier Sense Lost During Frame Transmission
  - When this error occurs and no collision is detected in the frame, the FEC sets the CSL bit in X_STATUS register. The frame is transmitted normally. No retries are performed as a result of this error.
  - No interrupt is generated as a result of this error
- Retransmission Attempts Limit Expired
  - When this error occurs, the FEC terminates transmission. All remaining buffers for that frame are then flushed and closed, and the RL bit is set in the X_STATUS register. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame.
  - The COL_RETRY_LIM interrupt is asserted if enabled in the ETH_IMASK register
- Late Collision
  - When a collision occurs after the slot time (512 bits starting at the Preamble), the FEC terminates transmission. All remaining buffers for that frame are then flushed and closed, and the LC bit is set in the X_STATUS register. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame.
  - The LATE_COL interrupt is asserted if enabled in the ETH_IMASK register
- Heartbeat
  - Some transceivers have a self-test feature called heartbeat or signal quality error. To signify a good self-test, the transceiver indicates a collision to the FEC within 20 clocks after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network, but is rather an indication that the transceiver seems to be functioning properly. This is called the heartbeat condition.
  - If the HBC bit is set in the ETH_X_CNTRL register and the heartbeat condition is not detected by the FEC after a frame transmission, a heartbeat error occurs. When this happens, the FEC closes the buffer, sets the HB bit in the X_STATUS register, and generates the HBERR interrupt if it is enabled.

14.6.12 Reception Errors

- Overrun Error
  - If the receive block has data to put into the receive FIFO and the receive FIFO is full, the FEC sets the OV bit in the receive status word. All subsequent data in the frame is discarded and subsequent frames may also be discarded until the receive FIFO is serviced by the DMA and space becomes available. At this point, the receive frame/status word is written into the FIFO with the OV bit set. This frame must be discarded by the driver.
- Non-Octet Error (Dribbling Bits)

— The Ethernet controller manages as many as 7 dribbling bits when the receive frame terminates non-octet aligned and it checks the CRC of the frame on the last octet boundary. If there is a CRC error, the frame non-octet aligned (no) error is reported in the receive BD. If there is no CRC error, no error is reported.

- CRC Error
  — When a CRC error occurs with no dribble bits, the FEC closes the buffer and sets the CR bit in the RxBD. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.
- Frame Length Violation
  — When the receive frame length exceeds MAX_FL bytes, the BABT interrupt is generated and the LG bit in the end-of-frame receive BD is set. The frame is not truncated (truncation occurs if the frame length exceeds 2047 bytes).
- Truncation
  — When the receive frame length exceeds 2047 bytes, the frame is truncated and the TR bit is set in the receive BD.

This page is intentionally left blank.

Chapter 15
General Purpose Timers (GPT)

15.1 Introduction

The MPC5125 provides two General Purpose Timer (GPT) modules. Each General Purpose Timer (GPT) provides eight independent timer channels that perform general purpose I/O (GPIO), input capture, output compare, pulse width modulation and internal CPU timer functions. An external I/O pin is associated with each Timer Channel. A separate 16-bit prescaler and 16-bit counter is associated with each timer channel, thus achieving a range of 32 bits (but only 16-bit resolution).

Eight general-purpose timer (GPT) pins are configurable for:

- Input capture
- Output compare
- Pulse width modulation (PWM) output
- Simple GPIO
- Internal CPU timer

15.1.1 Modes of Operation

15.1.1.1 Input Capture

In Input Capture mode, the I/O pin is an input. Two counters are used for each timer channel in this mode. The first counter is the Internal counter and the second is the up/down counter. After enabled, when specified capture event occurs (rising edge, falling edge, either edge, or pulse—two consecutive edges), the internal counter value is latched in the status register. If enabled, a CPU interrupt is generated. The input capture function has the following submodes, which are controlled by the ICM bits in the GPT Enable and Mode Select (GPT_MODE) register (see Section 15.2.1.1, “GPT0–7 Enable and Mode Select (GPT_MODE) Registers”).

- Normal input capture mode. Only internal counter is active.
- Up submode as well as normal input capture submode. Both up/down counter and internal counter are active.
- Updown submode as well as input capture submode. Both counters are active. A pair of timer channels are used to implement this mode.
- Rotary counter as well as input capture counter mode.Both counters are active. A pair of timer channels are used to implement this mode.
- When changing from one submode into another submode, the TIMER should be disabled first.

15.1.1.2 Output Compare

In Output Compare mode, the I/O pin is an Output. When enabled, the counters run until they reach the programmed Terminal Count value. At this point, the specified output event is generated (toggle, pulse high, or pulse low). If enabled, a CPU interrupt is generated.

15.1.1.3 PWM

In PWM mode, the I/O pin is an Output. The user can program period and width values to create an adjustable, repeating output waveform on the I/O pin. A CPU interrupt can be generated at the beginning of each PWM Period, at which time a new Width value can be loaded. The new Width value, which represents ON time, is automatically applied at the beginning of the next period. There is no interrupt at the beginning of the first PWM Period. This mode is suitable for PWM audio encoding. The GPT can support a combination of PWM outputs.

15.1.1.4 Simple GPIO

In Simple GPIO mode, the I/O pin operates as a GPIO pin. It can be specified as Input or Output, according to the programmable GPIO field. GPIO mode is mutually exclusive of Input Capture, Output Compare, and PWM modes.

15.1.1.5 CPU Timer

In CPU Timer mode, the I/O pin is not used. Once enabled, the counters run until they reach a programmed Terminal Count. When this occurs, an interrupt can be generated to the CPU. This timer mode can be used simultaneously with the Simple GPIO mode.

15.1.2 Detailed Signal Descriptions

Table 15-1 provides detailed descriptions of the external GPTimer signals.

Table 15-1. GPT External Signals—Detailed Signal Descriptions

Signal	I/O	Description	
GPT*n*[0:7]	I/O	GPTimer *n* 0–7. Each GPTimer module provides eight pins, numbered 0 through 7. Each pin can be individually set to act as input or output, according to application needs.	
		State Meaning	Asserted/Negated—Defined per application.
		Timing	Assertion/Negation —Inputs can be asserted completely asynchronously. Outputs are asynchronous to any externally visible clock

15.2 Memory Map and Register Definition

Each GPT timer channel uses four 32-bit registers. Table 15-2 shows the memory map for the local access registers. Within each GPTimer module, the registers are numbered to correspond to the pin for the respective timer channel.

Table 15-2. GPT memory map

Offset from GPT_BASE[1] GPT1: 0xFF40_0B00 GPT2: 0xFF40_5000	Register	Access[2]	Reset Value[3]	Section/Page
0x00	GPT0 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-391
0x04	GPT0 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-394
0x08	GPT0 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-395
0x0C	GPT0 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-396
0x10	GPT1 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-391
0x14	GPT1 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-394
0x18	GPT1 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-395
0x1C	GPT1 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-396
0x20	GPT2 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-391
0x24	GPT2 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-394
0x28	GPT2 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-395
0x2C	GPT2 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-396
0x30	GPT3 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-391
0x34	GPT3 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-394
0x38	GPT3 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-395
0x3C	GPT3 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-396
0x40	GPT4 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-391
0x44	GPT4 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-394
0x48	GPT4 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-395

Table 15-2. GPT memory map (Continued)

Offset from GPT_BASE[1] GPT1: 0xFF40_0B00 GPT2: 0xFF40_5000	Register	Access[2]	Reset Value[3]	Section/Page
0x4C	GPT4 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-396
0x50	GPT5 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-391
0x54	GPT5 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-394
0x58	GPT5 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-395
0x5C	GPT5 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-396
0x60	GPT6 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-391
0x64	GPT6 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-394
0x68	GPT6 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-395
0x6C	GPT6 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-396
0x70	GPT7 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-391
0x74	GPT7 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-394
0x78	GPT7 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-395
0x7C	GPT7 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-396
0x80–0xFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

3 In this column, the symbol “U” indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

15.2.1 Register Descriptions

15.2.1.1 GPT0–7 Enable and Mode Select (GPT_MODE) Registers

Address: Base + 0x00 (GPT0) Base + 0x40 (GPT4)
Base + 0x10 (GPT1) Base + 0x50 (GPT5)
Base + 0x20 (GPT2) Base + 0x60 (GPT6)
Base + 0x30 (GPT3) Base + 0x70 (GPT7)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OCPW										OCT		ICM		ICT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R				CE		STOP_ CONT	OPEN _DRN	INT EN			GPIO			TIMER_MS		
W	S[1]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-1. GPT0–GPT7 Enable and Mode Select (GPT_MODE) Registers

[1] Note: This bit must be set to 0 in normal working mode. Do not write 1 to this bit.

Table 15-3. GPT_MODE Field Descriptions

Field	Description
OCPW	Output Compare Pulse Width. Applies to OC Pulse types only. This field specifies the number of IP bus clocks (non-prescaled) to create a short output pulse at each Output Event. This pulse is generated at the end of the OC period and overlays the next OC period (rather than adding to the period).
OCT	Output Compare Type. Describes action to occur at each output compare event, as follows: 00 Special case, output is immediately forced low without respect to each output compare event. 01 Output pulse highs, initial value is low (OCPW field applies). 10 Output pulses low, initial value is high (OCPW field applies). 11 Output toggles. GPIO modalities can be used to achieve an initial output state prior to enabling OC mode. It is important to move directly from one GPIO output mode to another OC mode and not to pass through the Timer_MS=000 state. To prevent the Internal Timer Mode from engaging during the GPIO state, CE bit should be held low during the configuration steps. GPIO initialization is needed when presetting a Timer I/O to 1 in conjunction with a simple toggle OCT setting. **Note:** For Stop Mode operation (see the STOP_CONT bit in this table) it is necessary to pass through the mode_sel = 0 state to restart the output compare counters with their programmed values. See prescale and count fields in Section 15.1.1, “Modes of Operation.”

Table 15-3. GPT_MODE Field Descriptions (Continued)

Field	Description
ICM	Input Capture Mode—describes the input capture sub-mode as follows: 00 Normal input capture submode. 01 Up submode as well as normal input capture submode. Timer up/down counter increases one if it detects IC event. 10 Updown submode as well as input capture submode. Timers must be used in pairs. Timer 0 is paired with Timer 1, Timer 2 is paired with Timer 3, Timer 4 is paired with Timer 5 and Timer 6 is paired with Timer 7. For example, the timer0 up/down counter increases by one if timer0 detects IC event. The Timer 0 up/down counter decreases by one if timer1 detects an IC event. The timer 0 up/down counter remains unchanged if an IC event occurs on both channels during a single prescaled clock count. Timers 2 and 3, Timers 4 and 5, and Timers 6 and 7 operate in a similar fashion. 11 Rotary submode as well as input capture submode. When an IC event is detected on Timer 0, the Timer 0 up/down counter is incremented by 1 if Timer 1 is driven to a logic 0. the Timer 0 up/down counter decrements by 1 if Timer 1 is driven to a logic 1. Timers 2 and 3, Timers 4 and 5 and Timers 6 and 7 operate in a similar fashion. When up/down counter overflow or underflow occurs, a CPU interrupt can be generated if the interrupt enable bit is set. The IC event type is defined by the ICT field. **Note:** The up/down counter value can be read from the GPT0 – GPT7 Input and Up/Down Counter Output (GPT_COUNTER) Registers," bits 16-31. The value represents how many times an event happens. It is independent of the IC counter, which runs when enabled and latches the value when the IC event happens. Normally counter means IC counter (input capture counter) when timer is in Input Capture mode.When changing from one submode into another submode, the TIMER should be disabled.
ICT	Input Capture Type. Describes the input transition type required to trigger an input capture event, as follows: 00 Any input transition causes an IC event. 01 IC event occurs at input rising edge. 10 IC event occurs at input falling edge. 11 IC event occurs at any input pulse (i.e., at 2nd input edge). **Note:** For ICT 11 (pulse capture), status register records only the pulse width.
CE	Counter Enable. Bit enables or resets the internal counter during Internal timer modes only. CE must be high to enable these modes. If low, counter is held in reset. This bit is secondary to the timer mode select bits (Timer_MS). If the Timer_MS contains 1XX, the internal timer modes are enabled. CE can then enable or reset the internal counter without changing the Timer_MS field. GPIO operation is also available in this mode. 1 = enabled

Table 15-3. GPT_MODE Field Descriptions (Continued)

Field	Description
STOP_CONT	Stop Continuous—Applies to multiple modes, as follows: 0 Stop 1 Continuous **IC mode** • Stop operation—At each IC event, counter is reset. • Continuous operation—counter is not reset at each IC event. • Effect is to create Status count values that are cumulative between Capture events. If the Pulse Mode Capture mode is specified, the Stop_Cont bit is not used, operation fixed as if it were Stop. **OC mode** • Stop operation—Counter resets and stops at first OC event. **Note:** Software needs to pass through Timer_MS=000 state to restart timer. • Continuous operation—counter resets and continues at each OC event. • Effect to is create back-to-back periodic OC events. **PWM mode** • Bit not used, operation is always Continuous. **CPU Timer mode** • Stop operation—On counter expiration, Timer waits until Status bit is cleared by passing through Timer_MS=000 state before beginning a new cycle. • Continuous operation—On counter expiration, Timer resets and immediately begin a new cycle. • Effect is to generate fixed periodic timeouts. **GPIO modes** • Bit not used.
OPEN_DRN	Open Drain 0 Normal I/O 1 Open Drain emulation—affects all modes that drive the I/O pin (GPIO, OC, & PWM). Any output 1 is converted to a tri-state condition on the I/O pin.
INTEN	Interrupt enable. Enables interrupt generation to the CPU for all modes (IC, OC, PWM, and Internal Timer).
GPIO	GPIO mode type. Simple GPIO functionality that can be used simultaneously with the Internal Timer mode. It is not compatible with IC, OC, or PWM modes, since these modes require the usage of the I/O pin. 0x Timer enabled as simple GPIO input 10 Timer enabled as simple GPIO output, value=0 11 Timer enabled as simple GPIO output, value=1 (tri-state if Open_Drn=1) While in GPIO modes, internal timer mode is also available. To prevent undesired timer expiration, set the CE bit low.
TIMER_MS	Timer Mode Select (and module enable). 000 Timer module not enabled. Associated I/O pin is in input state. All Timer operation is completely disabled. Control and status registers remain accessible. This mode should be entered when timer is to be re-configured, except where the user does not want the I/O pin to become an input. 001 Timer enabled for input capture. Sub-mode can be set in field ICM. 010 Timer enabled for output compare. 011 Timer enabled for PWM. 1xx timer enabled for simple GPIO. Internal timer modes available. CE bit controls timer counter.

15.2.1.2 GPT0 – GPT7 Input and Up/Down Counter Output (GPT_COUNTER) Registers

Address: Base + 0x04 (GPT0), Base + 0x14 (GPT1), Base + 0x24 (GPT2), Base + 0x34 (GPT3), Base + 0x44 (GPT4), Base + 0x54 (GPT5), Base + 0x64 (GPT6), Base + 0x74 (GPT7)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRESCALE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	COUNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-2. GPT0–GPT7 Input and Up/Down Counter Output (GPT_COUNTER) Registers

Table 15-4. GPT_COUNTER field descriptions

Field	Description
PRESCALE	Prescale amount applied to internal counter (in IP bus clocks). **Note:** The prescale field should be written prior to enabling any timer mode. A prescale of 0x0001 means one IP bus clock per count increment. If prescale is 0 when any timer mode is started, it results in an effective prescale of 64 KB. The counter immediately begins and an output event occurs with the 64 KB prescale, rather than the desired value.
COUNT	Input Sets number of prescaled counts applied to reference events, as follows: OC—Number of prescaled counts counted before creating output event. PWM—Number of prescaled counts defining the PWM output period. Internal Timer—Number of prescaled counts counted before the timer expires. **Note:** Reading this register only returns the programmed value, intermediate values of the internal counter are not available to software. Output IC—When ICM is equal to 01/10/11, reading this field returns the internal up/down counter value. **Note:** Internal up/down counter starts at 0. Writing this field has no effect.

15.2.1.3 GPT0 – GPT7 PWM Configuration (GPT_PWM) Registers

Address: Base + 0x08 (GPT0) Base + 0x48 (GPT4)
Base + 0x18 (GPT1) Base + 0x58 (GPT5)
Base + 0x28 (GPT2) Base + 0x68 (GPT6)
Base + 0x38 (GPT3) Base + 0x78 (GPT7)
Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WIDTH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W								PWM OP								LOAD
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-3. GPT0–GPT7 PWM (GPT_PWM) Configuration Registers

Table 15-5. GPT_PWM field descriptions

Field	Description
WIDTH	PWM only. Defines ON time for output in prescaled counts. The PWM period is determined by the GPT_COUNTER register. ON time overlays the period time. If WIDTH = 0, output is always OFF. If WIDTH exceeds count value, output is always ON. ON and OFF polarity is set by the PWMOP bit.
PWMOP	Pulse Width Mode Output Polarity. Defines PWM output polarity for OFF time. Opposite state is ON time polarity. PWM cycles begin with ON time. 0 OFF TIME is a logic 0. 1 OFF TIME is a logic 1.
LOAD	Bit forces immediate period update. Bit auto clears itself. A new period begins immediately with the current count and width settings. 0 New count or width settings are not updated until end of current period. 1 New Count and Width settings take effect immediately. **Note:** Prescale setting is not part of this process. Changing prescale value while PWM is active causes unpredictable results for the period in which it was changed. The same is true for PWMOP bit.

15.2.1.4 GPT0–GPT7 Status (GPT_STATUS) Register

Address: Base + 0x0C (GPT0) Base + 0x4C (GPT4)
Base + 0x1C (GPT1) Base + 0x5C (GPT5)
Base + 0x2C (GPT2) Base + 0x6C (GPT6)
Base + 0x3C (GPT3) Base + 0x7C (GPT7)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CAPTURE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		OVF						PIN				UDOV	TEXP	PWMP	COMP	CAFT
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-4. GPT0–GPT7 Status (GPT_STATUS) Register

Table 15-6. GPT_STATUS field descriptions

Field	Description
CAPTURE	Read of internal counter latch at reference event. This is pertinent only in IC mode, in which case it represents the count value at the time the Input Event occurred. Capture status does not shadow the internal counter while an event is pending, it is updated only at the time the Input Event occurs. **Note:** If ICT is set to 11, which is Pulse Capture Mode, the Capture value records the width of the pulse. Also, the Stop_Cont bit is irrelevant in Pulse Capture Mode, operation is as if Stop_Cont were 0.
OVF	Represents how many times internal counter has rolled over. This is pertinent only during IC mode and would represent an extremely long period of time between Input Events. However, if STOP_CONT = 1 (indicating cumulative reporting of Input Events), this field could come into play. **Note:** This field is cleared by any sticky bit status write in the 5 bit fields below (27,28, 29, 30, 31).
PIN	Registered state of the I/O PIN (all modes). The IP Bus Clock registers the state of the I/O input. Valid, even if Timer is not enabled.
UDOV	Up/down counter has wrapped in up/updown/rotary IC submode , i.e. overflowed from 0xFFFF to 0x0000 or underflowed from 0x0000 to 0xFFFF. Cleared by writing 1 to this bit position. Also cleared if Timer_MS is 000 (i.e., Timer not enabled). See Note[1].
TEXP	Timer Expired in Internal Timer mode. Cleared by writing 1 to this bit position. Also cleared if Timer_MS is 000 (i.e., Timer not enabled). See Note[1].
PWMP	PWM end of period occurred. Cleared by writing 1 to this bit position. Also cleared if Timer_MS is 000 (i.e., Timer not enabled). See Note[1].
COMP	OC reference event occurred. Cleared by writing 1 to this bit position. Also cleared if Timer_MS is 000 (i.e., Timer not enabled). See Note[1].
CAFT	IC reference event occurred. Cleared by writing 1 to this bit position. Also cleared if Timer_MS is 000 (i.e., Timer not enabled). See Note[1].

[1] To clear any of these bits, it is necessary to clear **all** of them. An 1F must be written to bits 27:31.

15.3 Functional Description

The GPT provides eight independent timer channels that perform general purpose I/O, Input Capture, output compare, pulse width modulation and internal CPU timer functions. An external I/O pin is associated with each timer channel. A separate 16-bit prescaler and 16-bit internal counter is associated with each timer channel, thus achieving a range of 32 bits (but only 16-bit resolution). The 16-bit internal counter is not visible to the user. Thus, its value cannot be modified or read directly by software. In general, this 16-bit internal counter is used in one of two basic methods. The value of this counter can be captured upon the occurrence of some event. Then, the contents of the internal counter can be captured again at a second event. The difference between the two values is the number of prescaled clock counts between the two events. The second methodology involves adding a value to the internal counter and putting this value into a compare register. When the counter increments to this calculated value, a predetermined event can be programmed to occur.

An up/down counter is also implemented. This counter is visible to the user. In general, this counter increments in response to certain events and decrement in response to other events. The use of this counter is discussed below.

15.3.1 Input Capture Mode

In this mode, the timer I/O pin is an input. There are two counters used for each timer channel in this mode. The first counter is the Internal counter, and the second counter is the up/down counter. Once enabled, when the specified capture event occurs (rising edge, falling edge, either edge, or pulse—two consecutive edges), the internal counter value is latched in the status register. If enabled, a CPU interrupt is generated. The input capture function has the following submodes:

- Normal input capture mode
- Up mode
- Up down mode
- Rotary mode

These modes are controlled by the ICM bits described in Section 15.2.1, “Register Descriptions.”

15.3.1.1 Normal Input Capture Mode (IC MODE)

Only the internal counter is active in this mode. The IC Mode is selected by setting the ICM field of the GPT_MODE register associated with a particular timer channel to 0b00. In this mode, the ICT bits of the GPT_MODE are used to configure the Timer Channel pin as an input and to respond to any transition, a positive transition, a negative transition, or a pulse consisting of two consecutive edges. In the cases of any transition, positive transition or negative transition, the value of the internal counter is latched into the CAPTURE field of the GPT Status Register associated with the particular timer channel. If enabled by the INTEN bit of the GPT_MODE register for a particular channel, an interrupt to the CPU is generated.

15.3.1.2 UP Submodule

When a timer channel is programmed to use the UP Mode, both the up/down counter and internal counter are active. The up/down counter is incremented by one each time an Input Capture Event, as defined by

the ICT field of the GPT_MODE register, occurs. The value of the up/down counter can be obtained by reading the COUNT field of the GPT_COUNTER register for the particular timer channel.

An interrupt is generated, if enabled, when the UP Down Counter overflows.

The UP DOWN Counter cannot be modified by software.

15.3.1.3 UP DOWN Mode

When using the UP DOWN submode, the up/down counter and internal counter are active. A pair of GPT channels must be used to implement this mode. Timer Channel 0 is paired with Timer Channel 1, Timer Channel 2 is paired with Timer Channel 3, Timer Channel 4 is paired with Timer Channel 5 and Timer Channel 6 is paired with Timer Channel 7. The ICM field of the GPT_MODE register for both timer channels must be set to 0b10.

The ICT field in the GPT_MODE register for both channels must be programmed to detect the desired transitions. After a pair of channels is properly programmed, the up/down counter of one channel increments each time an Input Capture Event occurs on this channel of the pair and decrements by 1 when an Input Capture Event occurs on the other channel of the pair. That is, for the pair of channels consisting of Timer Channel 0 and Timer Channel 1, the up/down counter associated with Timer Channel 0 increments if an Input Capture event occurs on Timer Channel 0 and decrements if an Input Capture event occurs on Timer Channel 1.

An interrupt is generated, if enabled, when the up/down counter either underflows or overflows.

15.3.1.4 Rotary Mode

When using the ROTARY submode, both the up/down counter and internal counter are active. A pair of GPT channels must be used to implement this mode. Timer Channel 0 is paired with Timer Channel 1, Timer Channel 2 is paired with Timer Channel 3, Timer Channel 4 is paired with Timer Channel 5 and Timer Channel 6 is paired with Timer Channel 7. The ICM field of the GPT_MODE register for both timer channels must be set to 0b11.

The ICT field in the GPT_MODE register for one channel of the pair must be programmed to detect the desired transitions. After a pair of channels is properly programmed, the up/down counter increments each time an Input Capture Event occurs on one channel of the pair if the logic level on the input of the other Channel of the pair is a logic 0. The up/down counter decrements each time an Input Capture Event occurs on one channel of the pair if the logic level on the input of the other Channel of the pair is a logic 1.

For instance, if Timer Channel 0 is programmed to recognize positive transitions, the up/down counter associated with Timer Channel 0 increments each time a positive transition is detected on Timer Channel 0 if the logic level on Timer Channel 1 is a logic 0. The up/down counter associated with Timer Channel 0 decrements each time a positive transition is detected on Timer Channel 0 if the logic level on Timer Channel 1 is a logic 1.

15.3.2 Changing Sub-Modes

When using any mode of the GPT, unpredictable results can occur by arbitrarily switching from one mode to another. Before re-configuring a timer channel to a different mode of operation, the timer channel should be disabled by setting the TIMER_MS field in the GPT_MODE register to 0b000.

15.3.3 Output Compare

In this mode the I/O pin is an Output. When enabled the counters run until they reach the programmed Terminal Count value. At this point, the specified output event is generated (toggle, pulse hi, or pulse low). If enabled, a CPU interrupt is generated.

15.3.4 Force Output Low Immediately

The OCT field of the GPT_MODE register for a particular Timer Channel is written to 0b00 to force its associated Timer Channel Pin to a logic 0. No Output Compare event has to occur to force the Timer pin to a logic 0.

15.3.5 Output Pulse High

A GPT Timer Channel can be programmed to issue a single pulse with positive polarity in response to an Output Compare event. To enable this mode, the OCT field and the TIMER_MS field of the GPT_MODE register must be programmed to 01 and 010, respectively. The High Time of the pulse, specified in non-prescaled IP Bus clocks, is programmed into the OCPW field of the GPT_MODE register. To create an Output Compare Event, write a value to the COUNT field of the GPT_COUNTER register that specifies the time, expressed in prescaled IP Bus Clocks, when the next Output Compare event occurs. For example, if the OCPW field is written to 3, the PRESCALE field is written to 2, and the Count field is written to 4, the result is a positive pulse that is 3 IP Bus clocks wide that occurs 8 (Prescale = 2, Count = 4) IP Bus clocks after writing the TIMER_MS field of the GPT_MODE register to 0b010. If the STOP_CONT bit is set to 1 for continuous operation, the following wave form is generated.

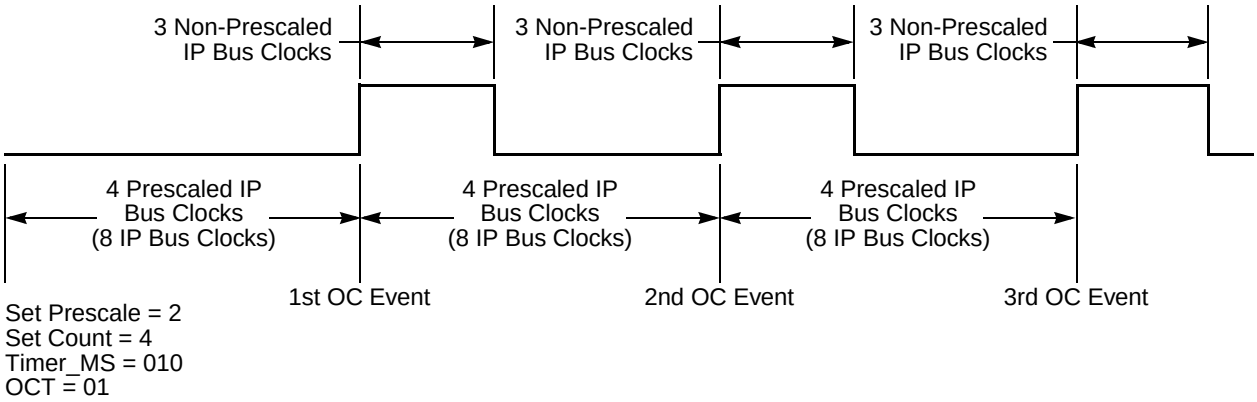


Figure 15-5. Output Pulse High Time Example

NOTE

It is the responsibility of the system software to set the Timer pin to the desired state before setting up the PRESCALE and COUNT fields in the GPT_COUNTER register.

15.3.6 Output Pulse Low

A GPT Timer Channel can be programmed to issue a single pulse with negative polarity in response to an Output Compare Event. To enable this mode, the OCT field and the TIMER_MS field of the GPT_MODE register must be programmed to 0b10 and 0b010, respectively. The LOW Time of the pulse, specified in non-prescaled IP Bus clocks, is programmed into the OCPW field of the GPT_MODE register. To create an Output Compare Event, write a value to the COUNT field of the GPT_COUNTER register that specifies the time, expressed in prescaled IP Bus Clocks, when the next Output Compare event occurs. For example, if the OCPW field is written to 3, the PRESCALE field is written to 2, and the Count field is written to 4, the result is a positive pulse which is 3 IP Bus clocks wide that occurs 8 (Prescale = 2, Count = 4) IP Bus clocks after writing the TIMER_MS field of the GPT_MODE register to 0b010. If the STOP_CONT bit is set to 1 for continuous operation, the following waveform is generated.

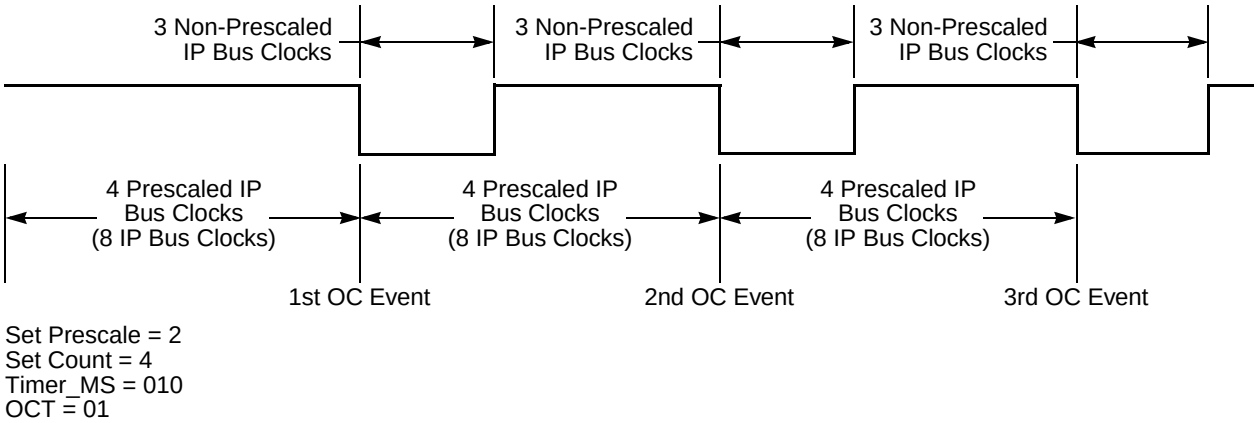


Figure 15-6. Output Pulse Low Time Example

NOTE

It is the responsibility of the system software to set the Timer pin to the desired state before setting up the PRESCALE and COUNT fields in the GPT_COUNTER register.

15.3.7 Output Toggle

A GPT Timer Channel can be programmed to toggle from its present value in response to an Output Compare Event. To enable this mode, the OCT field and the TIMER_MS field of the GPT_MODE register must be programmed to 0b11 and 0b010, respectively.

To create an Output Compare Event, a value can be written to the COUNT field of the GPT_COUNTER register that specifies the time, expressed in prescaled IP Bus Clocks, when the next Output Compare event occurs. If the PRESCALE field is written to 5 and the Count field is written to 6, the result is a transition at 30 (Prescale = 5, Count = 6) IP Bus clock intervals after setting the OCT field to 11 and the Timer_MS field of the GPT_MODE register to 010. If the STOP_CONT bit is set to 1 for continuous operation, the following wave form is generated. If the STOP_CONT bit is set to 0, only one Output compare Event occurs and the Timer Channel pin toggles only once.

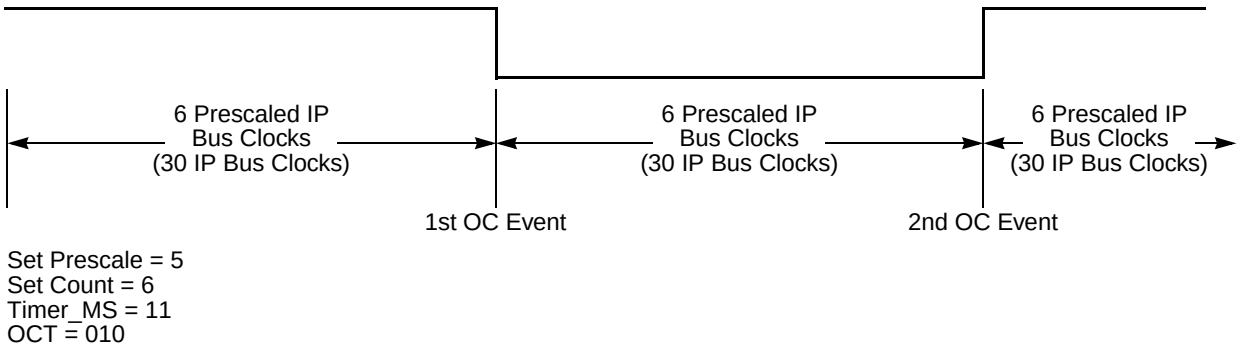


Figure 15-7. Output Compare Toggle Example

NOTE

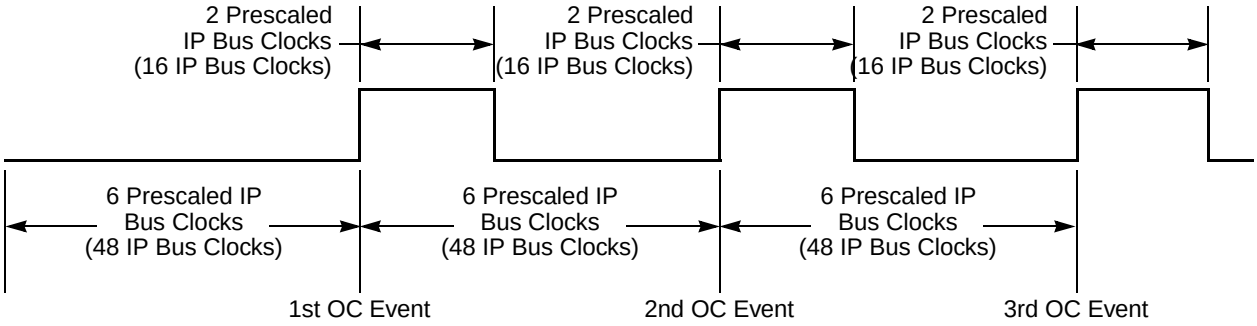
It is the responsibility of the system software to set the Timer pin to the desired state before setting up the PRESCALE and COUNT fields in the GPT_COUNTER register.

15.3.8 Pulse Width Modulation

In this mode the I/O pin is an Output. The user can program Period and Width values to create an adjustable, repeating output waveform on the I/O pin. A CPU interrupt can be generated at the beginning of each PWM Period, at which time a new Width value can be loaded. The new Width value, which represents ON time, is automatically applied at the beginning of the next period. There is no interrupt at the beginning of the first PWM Period. This mode is suitable for PWM audio encoding.

The ON TIME for the PWM signal is programmed into the WIDTH field of the GPT_COUNTER register in prescaled IP Bus Clocks. The Period of the PWM signal is programmed into the COUNT field of the GPT Counter Input Register. The PRESCALE field of the GPT Counter Input Register applies to both the COUNT value and the WIDTH Value. The ON TIME overlays the total period. That is, the WIDTH field determines the total period of the PWM signal.

In the following example, the Prescale field equals 8, the Count field equals 6 and the Width field equals 2.



Set Prescale = 8
Set Count = 6
Width = 2
Timer_MS = 011
PWMOP = 0

Figure 15-8. PWM Example with OFF TIME = LOW

NOTE

It is the responsibility of the system software to set the logic level on the TIMER pin to the desired value before involving the PWM mode.

NOTE

When the TIMER_MS field is set to 011 to start the PWM signal, the first period is at the logic level specified by the PWMOP bit. The active pulse whose width is specified by the WIDTH field overlays the following period.

NOTE

In the present example, it is assumed that the TIMER pin is at a logic 0 at the time that the TIMER_MS field is written. From a practical standpoint, this effectively means that the first active pulse occurs one full PWM period after writing the TIMER_MS field. If the TIMER pin is at a logic 1 at the time that the TIMER_MS field is written, it appears as though the first pulse is one full PWM period wide. Therefore, it is important to program the TIMER pin such that it is in a known state before writing to the TIMER_MS field.

The following example is similar to the example shown in Figure 15-8 with the exception that the OFF TIME is HIGH. In this case, PRESCALE = 9, COUNT = 4, WIDTH = 3, and PWMOP = 1.

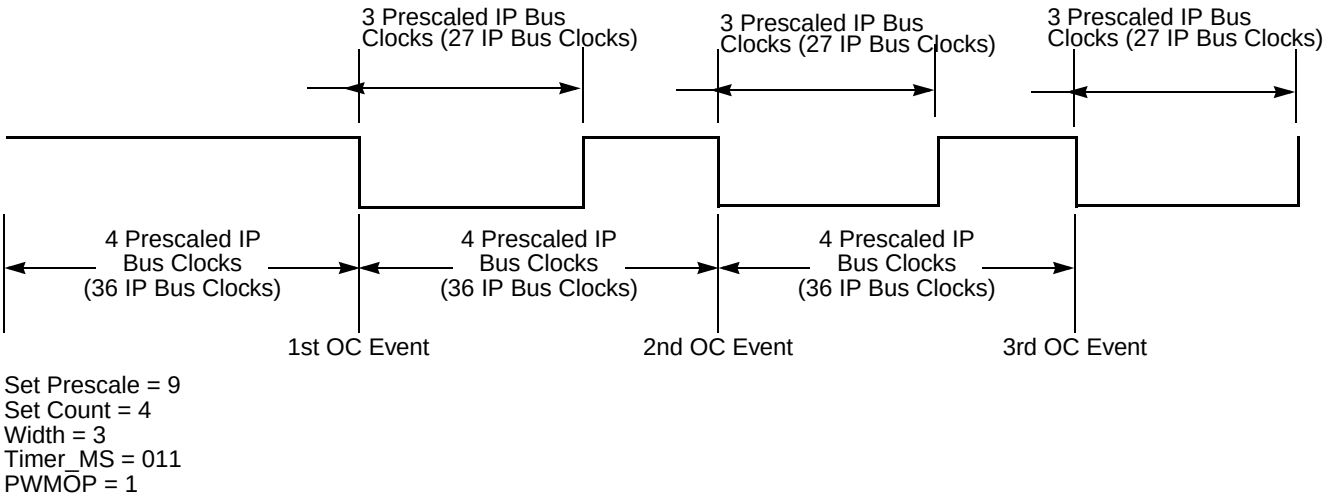


Figure 15-9. PWM Example with OFF TIME = HIGH

NOTE

It is the responsibility of the system software to set the logic level on the TIMER pin to the desired value before involving the PWM mode.

NOTE

When the TIMER_MS field is set to 011 to start the PWM signal, the first period is at the logic level specified by the PWMOP bit. The active pulse whose width is specified by the WIDTH field overlays the following period.

NOTE

In the present example, it is assumed that the TIMER pin is at a logic 1 at the time that the TIMER_MS field is written. From a practical standpoint, this effectively means that the first active pulse occurs one full PWM period after writing the TIMER_MS field. If the TIMER pin is at a logic 0 at the time that the TIMER_MS field is written, it appears as though the first pulse is one full PWM period wide. Therefore, it is important to program the TIMER pin such that it is in a known state before writing to the TIMER_MS field.

15.3.8.1 PWM Combination function

The GPT can support combinations of PWM outputs.

- Supports 4-bit LUT for all possible 2-bit input logic
- Provides the PWM output from single channel or any combinational logic output from eight input PWM signals
- Each output channel can be set individually

This function provides a look-up table (LUT) mechanism to generate any combinational logic signal output for eight PWM input signals. Its internal architecture is a tree fashion, which includes six LUTs. Each LUT has two inputs and one output. According to the values of two input signals, each LUT unit can

select one value (LUT[3], LUT[2], LUT[1], LUT[0]) as output signal from LUT configuration register. By setting the GPT control register, the GPT can also output any one of the LUT output signals or the original PWM signals.

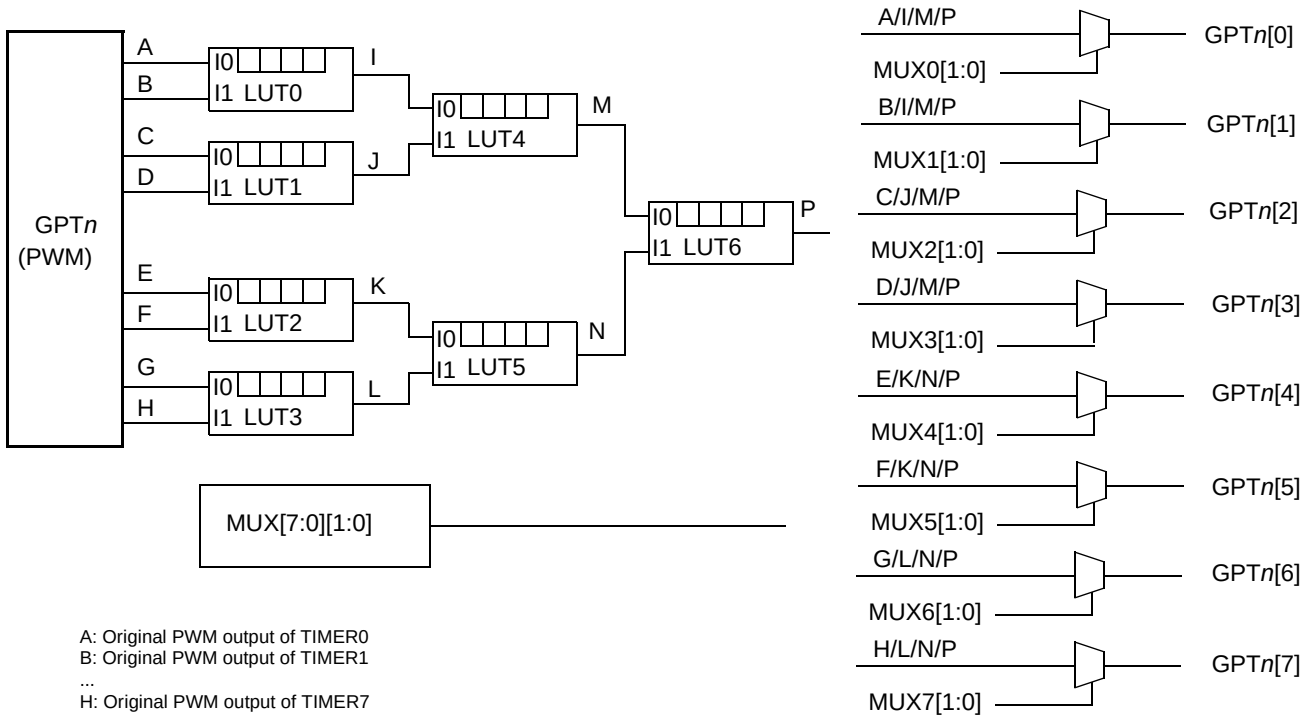


Figure 15-10. PWM combination function

15.3.9 Simple GPIO

In this mode, the I/O pin operates as a GPIO pin. Each Timer pin can individually be specified as Input or Output, according to the programmable GPIO field. GPIO mode is mutually exclusive of Input Capture, Output Compare and PWM modes. That is, in the GPIO mode, the TIMER pin cannot be used for input capture or to output a timer waveform. In GPIO mode, CPU Timer modes remain available.

15.3.9.1 CPU Timer

The I/O pin is not used in this mode. After enabled, the counters run until they reach a programmed Terminal Count. When this occurs, an interrupt can be generated to the CPU. This Timer mode can be used simultaneously with the Simple GPIO mode.

Chapter 16
General Purpose I/O (GPIO)

16.1 Introduction

This chapter describes the general purpose I/O module, including pin descriptions, register settings and interrupt capabilities.

There are two GPIO modules integrated into MPC5125, which supports 60 general-purpose I/O pins and four general input pins. Module GPIO1 provides pin GPIO00 to GPIO31, module GPIO2 provides pin GPIO32 to GPIO63. Pin GPIO00 to GPIO03 are four general input pins, and pin GPIO04 to GPIO63 are 60 general-purpose I/O pins. If a pin is configured as an input, it can optionally generate an interrupt upon detection of a change in state. If a pin is configured as an output, it can be configured as an open-drain output or a fully active output. When a GPIO00–GPIO30 pin is configured as an input, it can serve as a DMA request signal.

See Figure 16-1.

16.2 Features

The GPIO unit implements the following features:

- Thirty-two input/output pins
- All pins are configured as inputs when the MPC5125 reset signal is asserted
- Open-drain capability on all pin
- All pins, when configured as inputs, can optionally generate an interrupt upon detection of a change of state.
- When a GPIO pin is configured as an output, its DMA request functionality is disabled

The following sections provide an overview and detailed descriptions of the GPIO signals.

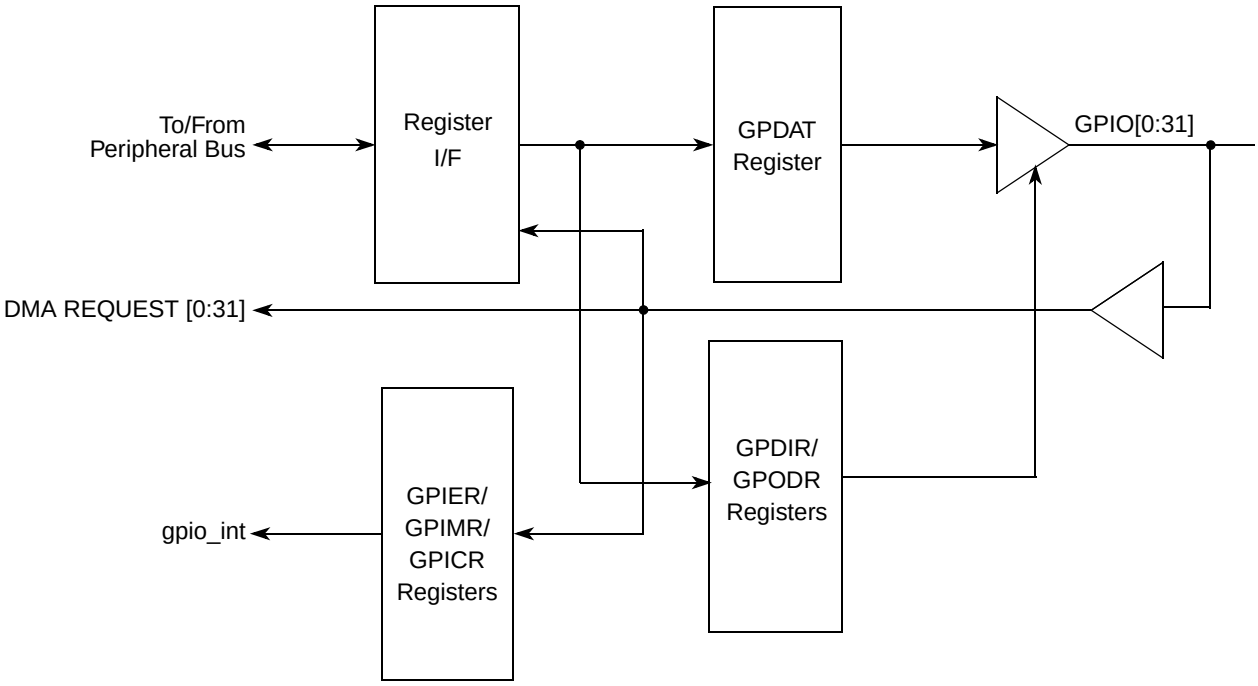


Figure 16-1. GPIO Module Block Diagram

16.3 Memory Map/Register Definition

All GPIO registers are 32 bits wide and are located on 32-bit address boundaries. All addresses used in this chapter are offsets from the GPIO module base address. A GPIO module memory map is shown in Table 16-1. Reading undefined portions of the memory map returns all zeros; writing has no effect.

Table 16-1. GPIO memory map

Offset from GPIO_BASE GPIO1: 0xFF40_1100 GPIO2: 0xFF40_1180[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x00	GPIO Direction Register (GPIO_GPDIR)	R/W	0x0000_0000	16.3.1.1/16-407
0x04	GPIO Open Drain Register (GPIO_GPODR)	R/W	0x0000_0000	16.3.1.2/16-408
0x08	GPIO Data Register (GPIO_GPDAT)	R/W	0x0000_0000	16.3.1.3/16-408
0x0C	GPIO Interrupt Event Register (GPIO_GPIER)	R/W	0x0000_0000	16.3.1.4/16-409
0x10	GPIO Interrupt Mask Register (GPIO_GPIMR)	R/W	0x0000_0000	16.3.1.5/16-410
0x14	GPIO External Interrupt Control Register 1 (GPIO_GPICR1)	R/W	0x0000_0000	16.3.1.6/16-410

Table 16-1. GPIO memory map (Continued)

Offset from GPIO_BASE GPIO1: 0xFF40_1100 GPIO2: 0xFF40_1180[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x18	GPIO External Interrupt Control Register 2 (GPIO_GPICR2)	R/W	0x0000_0000	16.3.1.6/16-410
0x1C–0x7F	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

3 In this column, the symbol “U” indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

16.3.1 Register Descriptions

16.3.1.1 GPIO Direction (GPIO_GPDIR) Register

The GPIO Direction (GPIO_GPDIR) register shown in Figure 16-2 defines the direction of individual GPIO pins.

NOTE

Bits D0–D31 of the GPIO_GPDIR register in GPIO1 set the I/O state of the GPIO 00–31 pins. Bits D0–D31 of GPIO_GPDIR register in GPIO2 set the I/O state of the GPIO 32–63pins.

Address: Base + 0x00 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30	D31
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-2. GPIO Direction Register (GPIO_GPDIR)

Table 16-2. GPIO_GPDIR field descriptions

Field	Description
D[0:31]	Direction. Indicates whether a pin is used as an input or an output. 0 The corresponding pin is an input. 1 The corresponding pin is an output. **Note:** For GPIO1, the D0–D3 bits have no meaning because the GPIO 00–03 pins are input only.

16.3.1.2 GPIO Open Drain Register (GPIO_GPODR)

The GPIO Open Drain Register (GPIO_GPODR) shown in Figure 16-3 defines the individual GPIO pins output drive structure.

NOTE

Bits D0–D31 of the GPIO_GPODR register in GPIO1 set the driver structure of the GPIO 00–31 pins. Bits D0–D31 of the GPIO_GPODR register in GPIO2 set the driver structure of the GPIO 32–63 pins.

Address: Base + 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30	D31
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-3. GPIO Open Drain Register (GPIO_GPODR)

Table 16-3. GPIO_GPODR field descriptions

Field	Description
D[0:31]	Output drive configuration. Indicates whether a pin is actively driven as an output or is an open-drain driver. 0 The I/O pin is actively driven as an output. 1 The I/O pin is an open-drain driver. As an output, the pin is driven active-low, otherwise it is tristated. **Note:** For GPIO1, the D0–D3 bit have no meaning because the GPIO 00–03 pins are input only.

16.3.1.3 GPIO Data (GPIO_GPDAT) Register

The GPIO Data (GPIO_GPDAT) register shown in Figure 16-4 carries the data in/out for individual GPIO pins.

NOTE

Bits D0–D31 of the GPIO_GPDAT register in GPIO1 are driven on the GPIO 00–31 pins when configured as outputs, and reflect the state of the GPIO 00–31 pins when configured as inputs.

Bits D0–D31 of the GPIO_GPDAT register in GPIO2 are driven on the GPIO 32–63 pins when configured as outputs, and reflect the state of the GPIO 32–63 pins when configured as inputs.

Address: Base + 0x08 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30	D31
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-4. GPIO Data Register (GPIO_GPDAT)

Table 16-4. GPIO_GPDAT field descriptions

Field	Description
D[0:31]	Data. Write data is latched and presented on external pins if GPIO_GPDIR has configured the GPIO pin as an output. Read operation always returns the data at the pin.

16.3.1.4 GPIO Interrupt Event Register (GPIO_GPIER)

The GPIO Interrupt Event Register (GPIO_GPIER) shown in Figure 16-5 carries information about the events that cause an interrupt. Each bit in the GPIO_GPIER register corresponds to an individual interrupt source. GPIO_GPIER bits are cleared by writing 1s. Writing 0 has no effect.

NOTE

Bits D0–D31 of the GPIO_GPIER register in GPIO1 are set in response to interrupt events occurring on the GPIO 00–31 pins. Bits D0–D31 of the GPIO_GPIER register in GPIO2 are set in response to interrupt events occurring on the GPIO 32–63 pins.

Address: Base + 0x0C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30	D31
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-5. GPIO Interrupt Event Register (GPIO_GPIER)

Table 16-5. GPIO_GPIER field descriptions

Field	Description
D[0:31]	Interrupt events. Indicates whether the interrupt event occurred on the corresponding GPIO pin. 0 No interrupt event occurred on the corresponding GPIO pin. 1 Interrupt event occurred on the corresponding GPIO pin.

16.3.1.5 GPIO Interrupt Mask Register (GPIO_GPIMR)

The GPIO Interrupt Mask Register (GPIO_GPIMR) shown in Figure 16-6 defines the interrupt masking for the individual GPIO pins. When a masked interrupt occurs, the corresponding GPIO_GPIER bit is set, regardless of the GPIO_GPIMR state. When one or more non-masked interrupt events occur, the GPIO module issues an interrupt to the Power Architecture core.

NOTE

Bits D0–D31 of the GPIO_GPIMR register in GPIO1 mask interrupts from the GPIO 00–31 pins. Bits D0–D31 of the GPIO_GPIMR register in GPIO2 mask interrupts from the GPIO 32–63 pins.

Address: Base + 0x10 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30	D31
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-6. GPIO Interrupt Mask Register (GPIO_GPIMR)

Table 16-6. GPIO_GPIMR field descriptions

Field	Description
D[0:31]	Interrupt mask. Indicates whether an interrupt event is masked or non-masked. 0 The input interrupt pin is masked (disabled). 1 The input interrupt pin is non-masked (enabled).

16.3.1.6 GPIO Interrupt Control Register 1 and 2 (GPIO_GPICR1 and GPIO_GPICR2)

The GPIO interrupt control register 1 (GPIO_GPICR1) and GPIO interrupt control register 2 (GPIO_GPICR2) shown in Figure 16-7 determines which type of event causes each individual GPIO pin

to set their associated bit in the GPIO interrupt event register and, if enabled, causes an interrupt to be asserted to the CPU.

NOTEThe D[0:31] bit fields of the GPIO interrupt control registers specify which type of event causes an interrupt for GPIO [0:31]. The interrupt function for each GPIO pin is individually programmable.

The D[0:31] bit fields of the GPIO_GPICR1 and GPIO_GPICR2 registers in GPIO1 specify the interrupt event type for GPIO pins 00–31. The D[0:31] bit fields of the GPIO_GPICR1 and GPIO_GPICR2 registers in GPIO2 specify the interrupt event type for GPIO pins 32–63.

Address: Base + 0x14 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	D0		D1		D2		D3		D4		D5		D6		D7	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	D8		D9		D10		D11		D12		D13		D14		D15	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-7. GPIO Interrupt Control Register 1 (GPIO_GPICR1)

Address: Base + 0x18 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	D16		D17		D18		D19		D20		D21		D22		D23	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	D24		D25		D26		D27		D28		D29		D30		D31	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-8. GPIO Interrupt Control Register (GPIO_GPICR2)

Table 16-7. GPIO_GPICR1 and GPIO_GPICR2 field descriptions

Field	Description
D[0:31]	Edge detection mode. The corresponding GPIO pin asserts an interrupt request according to the following: 00 Any change on the state of the GPIO pin generates an interrupt request. 01 Low-to-high change on the GPIO pin generates an interrupt request. 10 High-to-low change on the GPIO pin generates an interrupt request. 11 Pulse (any 2 transitions) on the GPIO pin generates an interrupt request.

16.4 Functional Description

The GPIO module supports 32 GPIO pins. Each GPIO pin can be configured as an input or output. If a GPIO pin is configured as an input, the pin can optionally generate an interrupt upon the detection of a change in state. If the GPIO interrupt control register bits for a particular GPIO pin are configured as 00, the GPIO pin detects any change of state, sets its corresponding bit in the GPIO_GPIER register and, if enabled, generates an interrupt. If the GPIO interrupt control register bits for a particular GPIO pin are configured as 01 or 10, the GPIO pin respectively detects a low-to-high transition or a high-to-low transition, sets the corresponding bit in the GPIO_GPIER register and, if enabled, generates an interrupt. If a GPIO pin is configured as an output, it can be individually configured as an open-drain or a fully active output.

Chapter 17
Integrated Programmable Interrupt Controller (IPIC)

17.1 Introduction

This chapter describes the IPIC interrupt protocol, various types of interrupt sources controlled by the IPIC unit, and the IPIC registers with some programming guidelines. The MPC5125 IPIC unit prioritizes and manages interrupts from the following controller units:

- DDR memory controller (DDR)
- LocalPlus controller (LPC)
- Two ethernet controllers (FEC)
- Ten programmable serial controllers (PSC)
- Two USB ULPI controllers (USB)
- NAND Flash controller (NFC)
- System bus arbiter (SBA)
- Real time clock timer (RTC)
- Sixteen Global Timers (GPT)
- Software Watchdog timer (WDT)
- I^2C controllers (I^2C)
- Controller Area Network (CAN)
- Byte Data Link controller (BDLC)
- Display Interface Unit (DIU)
- Secure Digital Host controller (SDHC)
- Direct Memory Access (DMA2)
- Power Management controller (PMC)
- General-purpose IO controller (GPIO)
- PSC FIFO controller (FIFOC)
- Temperature Sensor (TEMP)
- IC Identification (IIM)

The interrupt sources controlled by the IPIC unit cause exceptions in the processor core. The internal interrupt signal ($\overline{int}$) is the main interrupt output from the IPIC to the Power Architecture core and causes the regular interrupt exception. The $\overline{cint}$ signal is the critical interrupt output from the IPIC to the e300 core and causes the critical interrupt exception. The $\overline{smi}$ signal is the system management interrupt output from the IPIC to the Power Architecture core and causes the system management interrupt exception. The machine check exception is caused by the internal $\overline{mcp}$ signal generated by the IPIC, informing the host processor of error conditions, assertion of the external $\overline{IRQ0}$ machine-check request (enabled when IPIC_SEMSR[SIRQ0] = 1), and other conditions.

Figure 17-1shows the relationship of the various functional blocks and external signals of the MPC5125 to the IPIC unit.

The IPIC receives interrupt request signals from the sources external and internal to the integrated device

The unit selects the highest priority interrupt from all current interrupts and forwards it to the internal processor core.

The IPIC also manages an internal non-maskable machine-check processor signal ($\overline{mcp}$) and interrupt generated by the off-chip interrupt sources ($\overline{IRQ}$[1:0]).

The interrupt router of the IPIC monitors the outputs of the internal configuration registers. When the priority is highest in one of the received interrupt signals, the IPIC sets the corresponding bit in one of the interrupt pending registers (IPIC_SIPNR*x* or IPIC_SEPNR). If the interrupt is not masked, the IPIC asserts the $\overline{int}$, $\overline{cint}$, or $\overline{smi}$ signal to indicate an interrupt request to the processor. When the processor is executing the specific interrupt handler code, the processor must vectorize the external interrupt handler by explicitly reading (in software) the corresponding interrupt vector register (IPIC_SIVCR, IPIC_SCVCR or IPIC_SMVCR). In response to this read, the IPIC unit returns the vector (associated with the interrupt source) to the interrupt handler routine. In addition, the handler can vectorize different branches of interrupt handling.

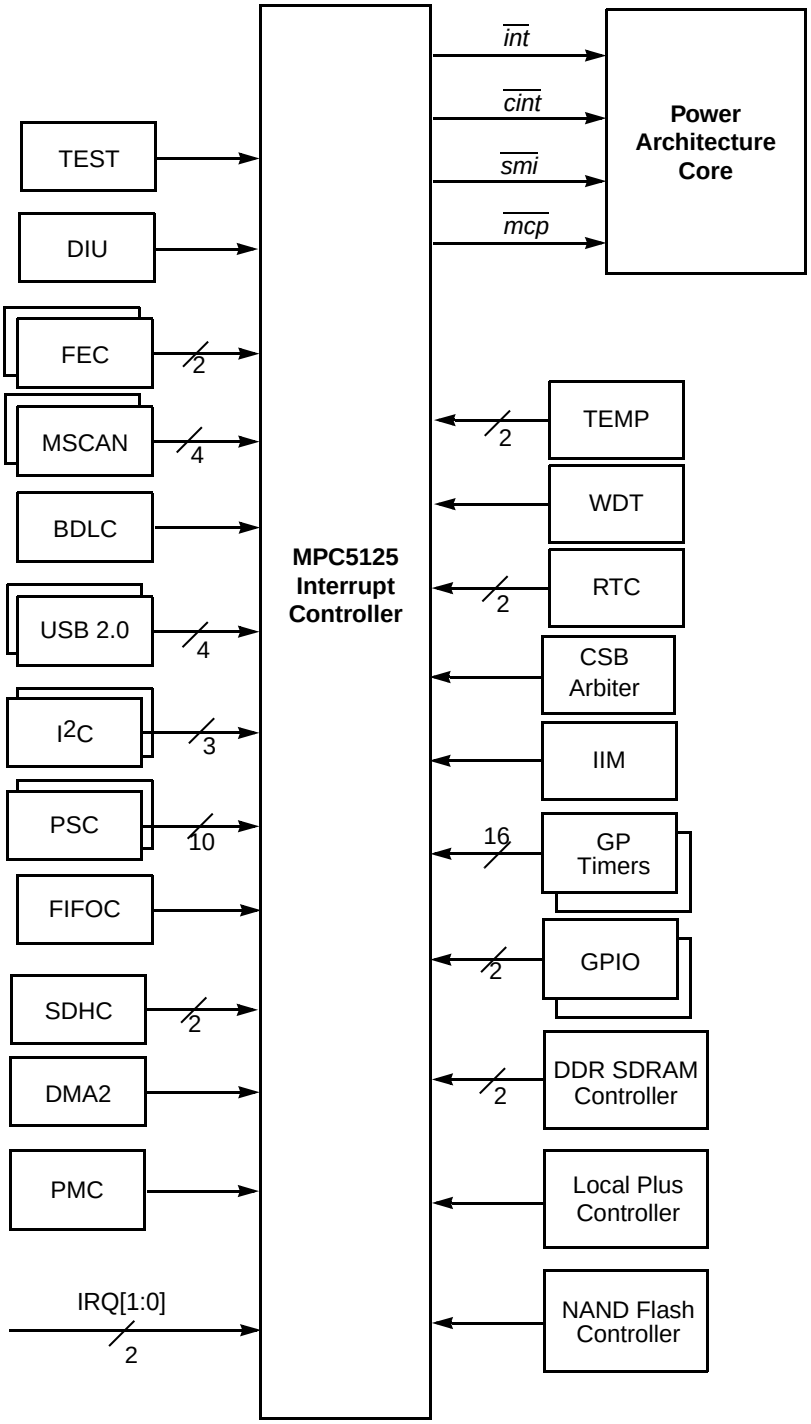


Figure 17-1. MPC5125 Interrupts Block Diagram

The IPIC can receive 62 separate interrupts from three different interrupt domains as follows:

- Two external interrupts — off-chip interrupt signals sources are $\overline{\text{IRQ}}$[1:0]

- Fifty-seven internal interrupts — on-chip interrupt signals sources are DDR, LPC, NFC, DMA, FEC, PSC, FIFOC, USB, CSB arbiter, CAN, BDLC, DIU, SDHC, RTC, I^2C, GPIO, GTM, TEMP, IIM and PMC.
- One external and three internal non-maskable machine check exceptions. Off-chip interrupt signal source is $\overline{\text{IRQ0}}$. On-chip MCP interrupt signals sources are software watchdog timer (WDT),TEMP and system bus arbiter (SBA)

The interrupt controller provides the ability to mask each interrupt source. Multiple events within GPIO or SBA peripheral event, are also maskable.

When the IPIC receives an internal or external interrupt, its configuration register is checked to determine if it should be serviced as a normal external interrupt by the processor core (through the $\overline{int}$ signal). As a third alternative, if the incoming interrupt has been configured as a critical or system management interrupt, the IPIC completes the processing of the interrupt by asserting $\overline{cint}$ or $\overline{smi}$ to the core. The assertion of the $\overline{cint}$ or $\overline{smi}$ signal to the core causes the interrupt to be serviced as a critical or a system management interrupt, respectively.

17.1.1 Overview

The interrupt controller provides interrupt management responsible for receiving hardware-generated interrupts from different sources (both internal and external). It also prioritizes and delivers the interrupts to the CPU for servicing.

17.1.2 Features

The IPIC unit implements the following features:

- Supports two external and internal discrete vectorized interrupt sources
- Supports one external and three internal machine check processor (MCP) interrupt sources
- Programmable highest priority request (can be programmed to support a critical ($\overline{cint}$) or system management ($\overline{smi}$) interrupt type)
- Two programmable priority mixed groups of four on-chip and four external interrupt signals with two priority schemes for each group: grouped and spread
- Four programmable priority internal groups of eight on-chip interrupt signals with two priority schemes for each group: grouped and spread
- Two highest priority interrupts from each group can be programmed to support a critical ($\overline{cint}$) or system management ($\overline{smi}$) interrupt type
- External and internal interrupts directed to host processor
- Unique vector number for each interrupt source

17.1.3 Modes of Operation

The IPIC unit can operate in the core enable or core disable mode.

17.1.3.1 Core Enable Mode

In core enable mode, all the MPC5125 internal interrupts and all machine check interrupts are routed towards the IPIC unit and from the IPIC the interrupts are sent to the Power Architecture core.

17.1.3.2 Core Disable Mode

In core disable mode all the MPC5125 internal interrupts are routed to and from the IPIC, but the IPIC will not route any interrupt to the Power Architecture core.

17.2 Memory Map/Register Definition

17.2.1 Register Summary

The IPIC programmable register map occupies 256 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All IPIC registers are 32 bits wide and they are located on 32-bit address boundaries. Software can perform byte, half-word or word accesses to any IPIC registers. All addresses used in this chapter are offsets from the IPIC base, as defined in Chapter 2, "System Configuration and Memory Map (XLBMEN + Mem Map)."

Table 17-1 shows the memory map of the IPIC unit.

Table 17-1. IPIC Register Address Map

Offset from IPIC_BASE (0xFF40_0C00)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x00	System Global Interrupt Configuration Register (IPIC_SICFR)	R/W	0x0000_0000	17.2.1.1/17-418
0x04	System Global Interrupt Vector Register (IPIC_SIVCR)	R	0x0000_0000	17.2.1.2/17-420
0x08	System Internal Interrupt Pending Register (IPIC_SIPNR_H)	R	0x0000_0000	17.2.1.3/17-423
0x0C	System Internal Interrupt Pending Register (IPIC_SIPNR_L)	R	0x0000_0000	17.2.1.3/17-423
0x10	System Internal Interrupt Group A Priority Register (IPIC_SIPRR_A)	R/W	0x0530_9770	17.2.1.4/17-426
0x14	System Internal Interrupt Group B Priority Register (IPIC_SIPRR_B)	R/W	0x0530_9770	17.2.1.5/17-427
0x18	System Internal Interrupt Group C Priority Register (IPIC_SIPRR_C)	R/W	0x0530_9770	17.2.1.6/17-428
0x1C	System Internal Interrupt Group D Priority Register (IPIC_SIPRR_D)	R/W	0x0530_9770	17.2.1.7/17-429
0x20	System Internal Interrupt Mask Register (IPIC_SIMSR_H)	R/W	0x0000_0000	17.2.1.8/17-430
0x24	System Internal Interrupt Mask Register (IPIC_SIMSR_L)	R/W	0x0000_0000	17.2.1.8/17-430
0x28	System Internal Interrupt Control Register (IPIC_SICNR)	R/W	0x0000_0000	17.2.1.9/17-433
0x2C	System External Interrupt Pending Register (IPIC_SEPNR)	R/W	0xU000_0000	17.2.1.10/17-434

Table 17-1. IPIC Register Address Map (continued)

Offset from IPIC_BASE (0xFF40_0C00)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x30	System Mixed Interrupt Group A Priority Register (IPIC_SMPRR_A)	R/W	0x0530_9770	17.2.1.11/17-435
0x34	System Mixed Interrupt Group B Priority Register (IPIC_SMPRR_B)	R/W	0x0530_9770	17.2.1.12/17-436
0x38	System External Interrupt Mask Register (IPIC_SEMSR)	R/W	0x0000_0000	17.2.1.13/17-437
0x3C	System External Interrupt Control Register (IPIC_SECNR)	R/W	0x0000_0000	17.2.1.14/17-438
0x40	System Error Status Register (IPIC_SERSR)	R/W	0x0000_0000	17.2.1.15/17-439
0x44	System Error Mask Register (IPIC_SERMR)	R/W	0xE080_0000	17.2.1.16/17-440
0x48	Reserved			
0x4C	System External Interrupt Polarity Check Register (IPIC_SEPCR)	R/W	0x0000_0000	17.2.1.17/17-441
0x50	System Internal Interrupt Force Register (IPIC_SIFCR_H)	R/W	0x0000_0000	17.2.1.18/17-441
0x54	System Internal Interrupt Force Register (IPIC_SIFCR_L)	R/W	0x0000_0000	17.2.1.18/17-441
0x58	System External Interrupt Force Register (IPIC_SEFCR)	R/W	0x0000_0000	17.2.1.19/17-444
0x5C	System Error Force Register (IPIC_SERFR)	R/W	0x0000_0000	17.2.1.20/17-445
0x60	System Critical Interrupt Vector Register (IPIC_SCVCR)	R	0x0000_0000	17.2.1.21/17-445
0x64	SysteM Management Interrupt Vector Register (IPIC_SMVCR)	R	0x0000_0000	17.2.1.22/17-446
0x68–0xFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

3 In this column, the symbol “U” indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

17.2.1.1 System Global Interrupt Configuration Register (IPIC_SICFR)

The System Global Interrupt Configuration Register (IPIC_SICFR), shown in Figure 17-2, defines the highest priority interrupt and whether interrupts are grouped or spread in the priority table. See Table 17-28 for more information.

Address: Base + 0x00 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	HPI							0	MPSB	MPSA	0	IPSD	IPSC	IPSB	IPSA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	HPIT		0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-2. System Global Interrupt Configuration Register (IPIC_SICFR)

Table 17-2. IPIC_SICFR field descriptions

Field	Description
HPI	Highest Priority Interrupt. Specifies the 7-bit unique interrupt number/vector (Table 17-4) of the single interrupt controller interrupt source advanced to the highest priority in the IPIC Priority table (Table 17-28). HPI can be modified dynamically.
MPSB	Mixed interrupts Priority Scheme for group B. Selects the relative MIXB priority scheme. It cannot be changed dynamically. 0 Grouped. The MIXBs are grouped by priority at the top of the table. 1 Spread. The MIXBs are spread by priority in the table.
MPSA	Mixed interrupts Priority Scheme for group A. Selects the relative MIXA priority scheme. It cannot be changed dynamically. 0 Grouped. The MIXAs are grouped by priority at the top of the table. 1 Spread. The MIXAs are spread by priority in the table.
IPSD	Internal interrupts Priority Scheme for group D. Selects the relative SYSD priority scheme. It cannot be changed dynamically. 0 Grouped. The SYSDs are grouped by priority at the top of the table. 1 Spread. The SYSDs are spread by priority in the table.
IPSC	Internal interrupts Priority Scheme for group C. Selects the relative SYSC priority scheme. It cannot be changed dynamically. 0 Grouped. The SYSCs are grouped by priority at the top of the table. 1 Spread. The SYSCs are spread by priority in the table.
IPSB	Internal interrupts Priority Scheme for group B. Selects the relative SYSB priority scheme. It cannot be changed dynamically. 0 Grouped. The SYSBs are grouped by priority at the top of the table. 1 Spread. The SYSBs are spread by priority in the table.
IPSA	Internal Interrupts Priority Scheme for Group A. Selects the relative SYSA priority scheme. It cannot be changed dynamically. 0 Grouped. The SYSAs are grouped by priority at the top of the table. 1 Spread. The SYSAs are spread by priority in the table.

Table 17-2. IPIC_SICFR field descriptions (continued)

Field	Description
HPIT	HPI Priority Position IPIC Output Interrupt Type. Defines type of the IPIC output interrupt signal (*int*, *cint*, or *smi*) asserts its request to the core in the HPI priority position. These bits cannot be changed dynamically. If S/W really wants to change it, it has to make sure the corresponding interrupt source is masked or it won't happen during the change. The definition of HPIT is as follows: 00 *int* request is asserted to the core for HPI. 01 *smi* request is asserted to the core for HPI. 10 *cint* request is asserted to the core for HPI. 11 Reserved.

17.2.1.2 System Global Interrupt Vector Register (IPIC_SIVCR)

The System Global Interrupt Vector Register (IPIC_SIVCR), shown in Figure 17-3, contains a 7-bit code (Table 17-3) representing the unmasked interrupt source of the highest priority level.

Address: Base + 0x04 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	IVEC						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-3. System Global Interrupt Vector Register (IPIC_SIVCR)

Table 17-3. IPIC_SIVCR field descriptions

Field	Description
IVEC	Interrupt Vector. Specifies a 7-bit unique number of the IPIC's highest priority interrupt source, pending to the core. When an interrupt request occurs, IPIC_SIVCR can be read. If there are multiple interrupt sources, IPIC_SIVCR latches the highest priority interrupt. **Note:** The value of SIVEC cannot change while it is being read. **Note:** While the upper 24 bits of this register are set to zero at the release of PORESET, they can and will change during device operation. Therefore, when using the value of the IVEC field, be sure to mask off the upper 24 bits of this register.

Table 17-4 shows the definition of IVEC.

NOTE

Interrupt vector numbers are assigned to each module and cannot be changed. For example, the SDHC2 module always returns 0x000_0011 as its interrupt vector regardless of its relative priority in the SYSC group.

Table 17-4. IVEC Field Values

Interrupt Number	Meaning	Interrupt Vector	Default Group programming
0	Error (No Interrupt)	0x000_0000	—
1	GPT10	0x000_0001	SYSC0 (Grouped)
2	GPT11	0x000_0010	SYSC1 (Grouped)
3	SDHC2	0x000_0011	SYSC2 (Grouped)
4	FEC1	0x000_0100	SYSC3 (Grouped)
5	FEC2	0x000_0101	SYSC4 (Grouped)
6	NFC	0x000_0110	SYSC5 (Grouped)
7	LPC	0x000_0111	SYSC6 (Grouped)
8	SDHC1	0x000_1000	SYSC7 (Grouped)
9	I2C1	0x000_1001	SYSD0 (Grouped)
10	I2C2	0x000_1010	SYSD1 (Grouped)
11	I2C3	0x000_1011	SYSD2 (Grouped)
12	MSCAN1	0x000_1100	SYSD3 (Grouped)
13	MSCAN2	0x000_1101	SYSD4 (Grouped)
14	BDLC	0x000_1110	SYSD5 (Grouped)
15	GPT0	0x000_1111	SYSD6 (Grouped)
16	GPT1	0x001_0000	SYSD7 (Grouped)
17	IRQ1	0x001_0001	MIXA5 (Grouped)
18	Reserved	0x001_0010	MIXA6 (Grouped)
19	Reserved	0x001_0011	MIXA7 (Grouped)
20	Reserved	0x001_0100	MIXB4 (Grouped)
21	Reserved	0x001_0101	MIXB5 (Grouped)
22	Reserved	0x001_0110	MIXB6 (Grouped)
23	Reserved	0x001_0111	MIXB7 (Grouped)
24	Reserved	0x001_1000	Fixed priority
25	Reserved	0x001_1001	Fixed priority
26	Reserved	0x001_1010	Fixed priority
27	Reserved	0x001_1011	Fixed priority
28	Reserved	0x001_1100	Fixed priority
29	Reserved	0x001_1101	Fixed priority
30	Reserved	0x001_1110	Fixed priority
31	Reserved	0x001_1111	Fixed priority
32	PSC4	0x010_0000	SYSA0 (Grouped)
33	PSC5	0x010_0001	SYSA1 (Grouped)
34	PSC6	0x010_0010	SYSA2 (Grouped)
35	PSC7	0x010_0011	SYSA3 (Grouped)

Table 17-4. IVEC Field Values (continued)

Interrupt Number	Meaning	Interrupt Vector	Default Group programming
36	PSC8	0x010_0100	SYSA4 (Grouped)
37	PSC9	0x010_0101	SYSA5 (Grouped)
38	GPT8	0x010_0110	SYSA6 (Grouped)
39	GPT9	0x010_0111	SYSA7 (Grouped)
40	FIFOC	0x010_1000	SYSB0 (Grouped)
41	Reserved	0x010_1001	SYSB1 (Grouped)
42	Reserved	0x010_1010	SYSB2 (Grouped)
43	USB2OTG1	0x010_1011	SYSB3 (Grouped)
44	USB2OTG2	0x010_1100	SYSB4 (Grouped)
45	Reserved	0x010_1101	SYSB5 (Grouped)
46	Reserved	0x010_1110	SYSB6 (Grouped)
47	Reserved	0x010_1111	SYSB7 (Grouped)
48	IRQ0	0x011_0000	MIXA4 (Grouped)
49	Reserved	0x011_0001	Reserved
50	Reserved	0x011_0010	Reserved
51	Reserved	0x011_0011	Reserved
52	Reserved	0x011_0100	Reserved
53	Reserved	0x011_0101	Reserved
54	Reserved	0x011_0110	Reserved
55	Reserved	0x011_0111	Reserved
56	Reserved	0x011_1000	Reserved
57	Reserved	0x011_1001	Reserved
58	Reserved	0x011_1010	Reserved
59	Reserved	0x011_1011	Reserved
60	Reserved	0x011_1100	Reserved
61	Reserved	0x011_1101	Reserved
62	Reserved	0x011_1110	Reserved
63	Reserved	0x011_1111	Reserved
64	DIU	0x100_0000	MIXA0
65	DMA2	0x100_0001	MIXA1
66	Reserved	0x100_0010	MIXA2
67	Reserved	0x100_0011	MIXA3
68	PSC0	0x100_0100	MIXB0
69	PSC1	0x100_0101	MIXB1
70	PSC2	0x100_0110	MIXB2
71	PSC3	0x100_0111	MIXB3

Table 17-4. IVEC Field Values (continued)

Interrupt Number	Meaning	Interrupt Vector	Default Group programming
72	GPT2	0x100_1000	Fixed priority
73	GPT3	0x100_1001	Fixed priority
74	GPT4	0x100_1010	Fixed priority
75	GPT5	0x100_1011	Fixed priority
76	GPT6	0x100_1100	Fixed priority
77	GPT7	0x100_1101	Fixed priority
78	GPIO1	0x100_1110	Fixed priority
79	RTC SEC	0x100_1111	Fixed priority
80	RTC ALARM	0x101_0000	Fixed priority
81	DDR	0x101_0001	Fixed priority
82	SBA	0x101_0010	Fixed priority
83	PMC	0x101_0011	Fixed priority
84	USB2OTG1 WKUP	0x101_0100	Fixed priority
85	USB2OTG2 WKUP	0x101_0101	Fixed priority
86	GPIO2	0x101_0110	Fixed priority
87	TEMP 105C	0x101_0111	Fixed priority
88	IIM	0x101_1000	Fixed priority
89	PRIOMON	0x101_1001	Fixed priority
90	MSCAN3	0x101_1010	Fixed priority
91	MSCAN4	0x101_1011	Fixed priority
92	GPT12	0x101_1100	Fixed priority
93	GPT13	0x101_1101	Fixed priority
95	GPT14	0x101_1110	Fixed priority
95	GPT15	0x101_1111	Fixed priority

17.2.1.3 System Internal Interrupt Pending Registers (IPIC_SIPNR_H and IPIC_SIPNR_L)

Each bit in the system internal interrupt pending registers (IPIC_SIPNR_H and IPIC_SIPNR_L), shown in Figure 17-4 and Figure 17-5, corresponds to an internal interrupt source. When an interrupt is received, the interrupt controller sets the corresponding IPIC_SIPNR*x* bit. When a pending interrupt is handled, the user clears the IPIC_SIPNR*x* bit by clearing the corresponding event register bit.

Address: Base + 0x08 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PSC4	PSC5	PSC6	PSC7	PSC8	PSC9	GPT8	GPT9	FIFOC	0	0	USB2 OTG1	USB2 OTG2	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPT10	GPT11	SDHC2	FEC1	FEC2	NFC	LPC	SDHC1	I2C1	I2C2	I2C3	MSCAN1	MSCAN2	BDLC	GPT0	GPT1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-4. System Internal Interrupt Pending Register High (IPIC_SIPNR_H)

Table 17-5. IPIC_SIPNR_H field descriptions

Field	Description
PSC4	PSC4 internal interrupt source. When an interrupt is received, the interrupt controller sets the corresponding IPIC_SIPNR*x* bit. When a pending interrupt is managed, clear the corresponding IPIC_SIPNR*x* bit. However, if an event register exists, the unmasked event register bits should be cleared instead, causing the IPIC_SIPNR*x* bit to be cleared. IPIC_SIPNR*x* bits are read-only. Writing to this register has no effect. **Note:** The IPIC_SIPNR*x* bit positions are not changed according to their relative priority.
PSC5	PSC5 internal interrupt source.
PSC6	PSC6 internal interrupt source.
PSC7	PSC7 internal interrupt source.
PSC8	PSC8 internal interrupt source.
PSC9	PSC9 internal interrupt source.
GPT8	GPT8 internal interrupt source.
GPT9	GPT9 internal interrupt source.
FIFOC	FIFOC internal interrupt source.
USB2OTG1	USB2OTG1 internal interrupt source.
USB2OTG2	USB2OTG2 internal interrupt source.
GPT10	GPT10 internal interrupt source.
GPT11	GPT11 internal interrupt source.
SDHC2	SDCH2 internal interrupt source.
FEC1	FEC1 internal interrupt source.
FEC2	FEC2 internal interrupt source.
NFC	NFC internal interrupt source.
LPC	LPC internal interrupt source.
SDHC1	SDHC1 internal interrupt source.

Table 17-5. IPIC_SIPNR_H field descriptions (continued)

Field	Description
I2C1	I2C1 internal interrupt source.
I2C2	I2C2 internal interrupt source.
I2C3	I2C3 internal interrupt source.
MSCAN1	MSCAN1 internal interrupt source.
MSCAN2	MSCAN2 internal interrupt source.
BDLC	BDLC internal interrupt source.
GPT0	GPT0 internal interrupt source.
GPT1	GPT1 internal interrupt source.

Address: Base + 0x0C Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DIU	DMA 2	0	0	PSC0	PSC1	PSC2	PSC3	GPT2	GPT3	GPT4	GPT5	GPT6	GPT7	GPIO 1	RTC SEC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RTC ALARM	DDR	SBA	PMC	USB2 OTG1 WKUP	USB2 OTG2 WKUP	GPIO 2	TEMP 105C	IIM	PRIO MON	MSC AN3	MSC AN4	GPT12	GPT13	GPT14	GPT15
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-5. System Internal Interrupt Pending Register Low (IPIC_SIPNR_L)

Table 17-6. IPIC_SIPNR_L field descriptions

Field	Description
DIU	DIU internal interrupt source. When an interrupt is received, the interrupt controller sets the corresponding IPIC_SIPNR*x* bit. When a pending interrupt is managed, clear the corresponding IPIC_SIPNR*x* bit. However, if an event register exists, the unmasked event register bits should be cleared instead, causing the IPIC_SIPNR*x* bit to be cleared. IPIC_SIPNR bits*x* are read-only. Writing to this register has no effect. **Note:** The IPIC_SIPNR*x* bit positions are not changed according to their relative priority.
DMA2	DMA2 internal interrupt source.
PSC0	PSC0 internal interrupt source.
PSC1	PSC1 internal interrupt source.
PSC2	PSC2 internal interrupt source.
PSC3	PSC3 internal interrupt source.
GPT2	GPT2 internal interrupt source.

Table 17-6. IPIC_SIPNR_L field descriptions (continued)

Field	Description
GPT3	GPT3 internal interrupt source.
GPT4	GPT4 internal interrupt source.
GPT5	GPT5 internal interrupt source.
GPT6	GPT6 internal interrupt source.
GPT7	GPT7 internal interrupt source.
GPIO1	GPIO1 internal interrupt source.
RTC SEC	RTC SEC internal interrupt source.
RTC ALARM	RTC ALARM internal interrupt source.
DDR	DDR internal interrupt source.
SBA	SBA internal interrupt source.
PMC	PMC internal interrupt source.
USB2OTG1 WKUP	USB2OTG1 WKUP internal interrupt source.
USB2OTG2 WKUP	USB2OTG2 WKUP internal interrupt source.
GPIO2	GPIO2 internal interrupt source.
TEMP 105C	TEMP 105C internal interrupt source.
IIM	IIM internal interrupt source.
PRIOMON	PRIOMON internal interrupt source.
MSCAN3	MSCAN3 internal interrupt source.
MSCAN4	MSCAN4 internal interrupt source.
GPT12	GPT12 internal interrupt source.
GPT13	GPT13 internal interrupt source.
GPT14	GPT14 internal interrupt source.
GPT15	GPT15 internal interrupt source.

17.2.1.4 System Internal Interrupt Group A Priority Register (IPIC_SIPRR_A)

The system internal interrupt group A priority register (IPIC_SIPRR_A), shown in Figure 17-6, defines the priority between the PSC4, PSC5, PSC6, PSC7, PSC8, PSC9, GPT8, and GPT9 internal interrupt signals.

Address: Base + 0x10 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYSA0P				SYSA1P				SYSA2P				SYSA3P			
W													0	0	0	0
Reset	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SYSA4P				SYSA5P				SYSA6P				SYSA7P			
W													0	0	0	0
Reset	1	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0

Figure 17-6. System Internal Interrupt Group A Priority Register (IPIC_SIPRR_A)

Table 17-7. IPIC_SIPRR_A field descriptions

Field	Description
SYSA0P	SYSA0 Priority Order. Defines which interrupt source asserts its request in the SYSA0 priority position. Do not program the same code to more than one priority position (0–7). These bits can be changed dynamically. The definition of SYSA0P is shown as follows: 000 PSC4 asserts its request in the SYSA0 position. 001 PSC5 asserts its request in the SYSA0 position. 010 PSC6 asserts its request in the SYSA0 position. 011 PSC7 asserts its request in the SYSA0 position. 100 PSC8 asserts its request in the SYSA0 position. 101 PSC9 asserts its request in the SYSA0 position. 110 GPT8 asserts its request in the SYSA0 position. 111 GPT9 asserts its request in the SYSA0 position.
SYSA1P–SYSA7P	Same as SYSA0P, but for SYSA1P–SYSA7P.

17.2.1.5 System Internal Interrupt Group B Priority Register (IPIC_SIPRR_B)

The System Internal Interrupt Group B Priority Register (IPIC_SIPRR_B), shown in Figure 17-7, defines the priority between the FIFOC, USB2OTG1, and USB2OTG2 internal interrupt signals.

Address: Base + 0x14 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYSB0P				SYSB1P				SYSB2P				SYSB3P			
W													0	0	0	0
Reset	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SYSA4P				SYSA5P				SYSB6P				SYSB7P			
W													0	0	0	0
Reset	1	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0

Figure 17-7. System Internal Interrupt Group B Priority Register (IPIC_SIPRR_B)

Table 17-8. IPIC_SIPRR_B field descriptions

Field	Description
SYSB0P	SYSB0 Priority order. Defines which interrupt source asserts its request in the SYSB0 priority position. The user should not program the same code to more than one priority position (0–7). These bits can be changed dynamically. The definition of SYSB0P is shown as follows: 000 FIFOC asserts its request in the SYSB0 position. 001 Reserved. 010 Reserved. 011 USB2OTG1 asserts its request in the SYSB0 position. 100 USB2OTG2 asserts its request in the SYSB0 position. 101 Reserved. 110 Reserved. 111 Reserved.
SYSB1P–SYSB7P	Same as SYSB0P, but for SYSB1P–SYSB7P.

17.2.1.6 System Internal Interrupt Group C Priority Register (IPIC_SIPRR_C)

The System Internal Interrupt Group C Priority Register (IPIC_SIPRR_C), shown in Figure 17-8, defines the priority between the GPT10, GPT11, SDHC2, FEC1, FEC2, NFC, LPC, and SDHC1 internal interrupt signals.

Address: Base + 0x18 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYSC0P				SYSC1P				SYSC2P				0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SYSC4P				SYSC5P				SYSC6P				0	0	0	0
W																
Reset	1	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0

Figure 17-8. System Internal Interrupt Group C Priority Register (IPIC_SIPRR_C)

Table 17-9. IPIC_SIPRR_C field descriptions

Field	Description
SYSC0P	SYSC0 Priority order. Defines which interrupt source asserts its request in the SYSC0 priority position. The user should not program the same code to more than one priority position (0–7). These bits can be changed dynamically. The definition of SYSC0P is shown as follows: 000 GPT10 asserts its request in the SYSC0 position. 001 GPT11 asserts its request in the SYSC0 position. 010 SDHC2 asserts its request in the SYSC0 position. 011 FEC1 asserts its request in the SYSC0 position. 100 FEC2 asserts its request in the SYSC0 position. 101 NFC asserts its request in the SYSC0 position. 110 LPC asserts its request in the SYSC0 position. 111 SDHC1 asserts its request in the SYSC0 position.
SYSC1P–SYSC7P	Same as SYSC0P, but for SYSC1P–SYSC7P.

17.2.1.7 System Internal Interrupt Group D Priority Register (IPIC_SIPRR_D)

The system internal interrupt group D priority register (IPIC_SIPRR_D), shown in Figure 17-9, defines the priority between the I2C1, I2C2, I2C3, MSCAN1, MSCAN2, BDLC, GPT0, and GPT1 internal interrupt signals.

Address: Base + 0x1C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYSD0P				SYSD1P				SYSD2P				0	0	0	0
W																
Reset	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SYSD4P				SYSD5P				SYSD6P				0	0	0	0
W																
Reset	1	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0

Figure 17-9. System Internal Interrupt Group D Priority Register (IPIC_SIPRR_D)

Table 17-10. IPIC_SIPRR_D field descriptions

Field	Description
SYSD0P	SYSD0 Priority order. Defines which interrupt source asserts its request in the SYSD0 priority position. The user should not program the same code to more than one priority position (0–7). These bits can be changed dynamically. The definition of SYSD0P is shown as follows: 000 I2C1 asserts its request in the SYSD0 position. 001 I2C2 asserts its request in the SYSD0 position. 010 I2C3 asserts its request in the SYSD0 position. 011 MSCAN1 asserts its request in the SYSD0 position. 100 MSCAN2 asserts its request in the SYSD0 position. 101 BDLC asserts its request in the SYSD0 position. 110 GPT0 asserts its request in the SYSD0 position. 111 GPT1 asserts its request in the SYSD0 position.
SYSD1P–SYSD7P	Same as SYSD0P, but for SYSD1P–SYSD7P.

17.2.1.8 System Internal Interrupt Mask Register (IPIC_SIMSR_H and IPIC_SIMSR_L)

Each implemented bit in the IPIC_SIMSR_H and IPIC_SIMSR_L, shown in Figure 17-10 and Figure 17-11, corresponds to an internal interrupt source. The user masks an interrupt by clearing the corresponding IPIC_SIMSR*x* bit. When an interrupt request occurs, the corresponding IPIC_SIPNR*x* bit is set, regardless of the IPIC_SIMSR*x* bit. However, if the corresponding IPIC_SIMSR*x* bit is cleared, no interrupt request is passed to the core.

When the IPIC_SIMSR*x* bit is cleared by the user at the same time an interrupt source requests an interrupt service, the request stops. If the user sets the IPIC_SIMSR*x* bit later, a previously pending interrupt request is processed by the core according to its assigned priority.

Address: Base + 0x20 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PSC4	PSC5	PSC6	PSC7	PSC8	PSC9	GPT8	GPT9	FIFOC	0	0	USB2OTG1	USB2OTG2	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPT10	GPT11	SDHC2	FEC1	FEC2	NFC	LPC	SDHC1	I2C1	I2C2	I2C3	MSCAN1	MSCAN2	BDLC	GPT0	GPT1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-10. System Internal Interrupt Mask High Register (IPIC_SIMSR_H)

Table 17-11. IPIC_SIMSR_H field descriptions

Field	Description
PSC4	PSC4 external interrupt source. Mask an interrupt by clearing the IPIC_SIMSR*x* bit. An interrupt can be enabled by setting the corresponding IPIC_SIMSR*x* bit. The IPIC_SIMSR*x* can be read at any time. **Note:** • IPIC_SIMSR*x* bit positions are not changed according to their relative priority. • Pending register bits set by multiple interrupt events can be cleared only by clearing all unmasked events in the corresponding event register. • If an IPIC_SIMSR*x* bit is masked at the same time the corresponding IPIC_SIPNR*x* bit causes an interrupt request to the core, the error vector is issued (if no other interrupts pending). Therefore, always include an error.
PSC5	PSC5 external interrupt source.
PSC6	PSC6 external interrupt source.
PSC7	PSC7 external interrupt source.
PSC8	PSC8 external interrupt source.
PSC9	PSC9 external interrupt source.
GPT8	GPT8 external interrupt source.
GPT9	GPT9 external interrupt source.
FIFOC	FIFOC external interrupt source.
USB2OTG1	USB2OTG1 external interrupt source.
USB2OTG2	USB2OTG2 external interrupt source.
GPT10	GPT10 external interrupt source.
GPT11	GPT11 external interrupt source.
SDHC2	SDCH2 external interrupt source.
FEC1	FEC1 external interrupt source.
FEC2	FEC2 external interrupt source.
NFC	NFC external interrupt source.
LPC	LPC external interrupt source.
SDHC1	SDHC1 external interrupt source.
I2C1	I2C1 external interrupt source.
I2C2	I2C2 external interrupt source.
I2C3	I2C3 external interrupt source.
MSCAN1	MSCAN1 external interrupt source.
MSCAN2	MSCAN2 external interrupt source.
BDLC	BDLC external interrupt source.
GPT0	GPT0 external interrupt source.
GPT1	GPT1 external interrupt source.

Address: Base + 0x24 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DIU	DMA2	0	0	PSC0	PSC1	PSC2	PSC3	GPT2	GPT3	GPT4	GPT5	GPT6	GPT7	GPIO1	RTC SEC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RTC ALARM	DDR	SBA	PMC	USB2 OTG1 WKUP	USB2 OTG2 WKUP	GPIO2	TEMP 105C	IIM	PRIO MON	MSC AN3	MSC AN4	GPT12	GPT13	GPT14	GPT15
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-11. System Internal Interrupt Mask Low Register (IPIC_SIMSR_L)

Table 17-12. IPIC_SIMSR_L field descriptions

Field	Description
DIU	DIU external interrupt source. Mask an interrupt by clearing the corresponding IPIC_SIMSR*x* bit. An interrupt can be enabled by setting the corresponding IPIC_SIMSR*x* bit. The IPIC_SIMSR*x* can be read at any time. **Note:** • IPIC_SIMSR*x* bit positions are not changed according to their relative priority. • Pending register bits that were set by multiple interrupt events can be cleared only by clearing all unmasked events in the corresponding event register. • If an IPIC_SIMSR*x* bit is masked at the same time the corresponding IPIC_SIPNR*x* bit causes an interrupt request to the core, the error vector is issued (if no other interrupts pending). Therefore, always include an error.
DMA2	DMA2 external interrupt source.
PSC0	PSC0 external interrupt source.
PSC1	PSC1 external interrupt source.
PSC2	PSC2 external interrupt source.
PSC3	PSC3 external interrupt source.
GPT2	GPT2 external interrupt source.
GPT3	GPT3 external interrupt source.
GPT4	GPT4 external interrupt source.
GPT5	GPT5 external interrupt source.
GPT6	GPT6 external interrupt source.
GPT7	GPT7 external interrupt source.
GPIO1	GPIO1 external interrupt source.
RTC SEC	RTC SEC external interrupt source.
RTC ALARM	RTC ALARM external interrupt source.
DDR	DDR external interrupt source.
SBA	SBA external interrupt source.
PMC	PMC external interrupt source.

Table 17-12. IPIC_SIMSR_L field descriptions (continued)

Field	Description
USB2OTG1 WKUP	USB2OTG1 WKUP external interrupt source.
USB2OTG2 WKUP	USB2OTG2 WKUP external interrupt source.
GPIO2	GPIO2 external interrupt source.
TEMP 105C	TEMP 105C external interrupt source.
IIM	IIM external interrupt source.
PRIOMON	PRIOMON external interrupt source.
MSCAN3	MSCAN3 external interrupt source.
MSCAN4	MSCAN4 external interrupt source.
GPT12	GPT12 external interrupt source.
GPT13	GPT13 external interrupt source.
GPT14	GPT14 external interrupt source.
GPT15	GPT15 external interrupt source.

17.2.1.9 System Internal Interrupt Control Register (IPIC_SICNR)

The System Internal Interrupt Control Register (IPIC_SICNR), shown in Figure 17-12, defines the IPIC output interrupt type (*int*, *cint*, or *smi*) in the SYSA0-SYSA1, SYSB0-SYSB1, SYSC0-SYSC1, and SYSD0-SYSD1 priority positions. All other priority positions assert an INT signal to the core.

Address: Base + 0x28 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYSD0T		SYSD1T		0	0	0	0	SYSC0T		SYSC1T		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SYSB0T		SYSB1T		0	0	0	0	SYSA0T		SYSA1T		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-12. System Internal Interrupt Control Register (IPIC_SICNR)

Table 17-13. IPIC_SICNR field descriptions

Field	Description
SYSD*x*T	SYSD*x* priority position IPIC output interrupt Type. Defines which type of IPIC output interrupt ($\overline{int}$, $\overline{cint}$, or $\overline{smi}$) asserts its request to the core in the SYSD*x* priority position. These bits cannot be changed dynamically. (to change it, software must make sure the corresponding interrupt source is masked or it cannot happen during the change). The definition of SYSD*x*T is as follows: 00 $\overline{int}$ request is asserted to the core for SYSD*x*. 01 $\overline{smi}$ request is asserted to the core for SYSD*x*. 10 $\overline{cint}$ request is asserted to the core for SYSD*x*. 11 Reserved.
SYSC*x*T	SYSC*x* priority position IPIC output interrupt Type. Defines which type of IPIC output interrupt ($\overline{int}$, $\overline{cint}$, or $\overline{smi}$) asserts its request to the core in the SYSC*x* priority position. These bits cannot be changed dynamically. (to change it, software must make sure the corresponding interrupt source is masked or it cannot happen during the change). The definition of SYSC*x*T is as follows: 00 $\overline{int}$ request is asserted to the core for SYSC*x*. 01 $\overline{smi}$ request is asserted to the core for SYSC*x*. 10 $\overline{cint}$ request is asserted to the core for SYSC*x*. 11 Reserved.
SYSB*x*T	SYSB*x* priority position IPIC output interrupt Type. Defines which type of IPIC output interrupt ($\overline{int}$, $\overline{cint}$, or $\overline{smi}$) asserts its request to the core in the SYSB*x* priority position. These bits cannot be changed dynamically. (to change it, software must make sure the corresponding interrupt source is masked or it cannot happen during the change). The definition of SYSB*x*T is as follows: 00 $\overline{int}$ request is asserted to the core for SYSB*x*. 01 $\overline{smi}$ request is asserted to the core for SYSB*x*. 10 $\overline{cint}$ request is asserted to the core for SYSB*x*. 11 Reserved.
SYSA*x*T	SYSA*x* priority position IPIC output interrupt Type. Defines which type of IPIC output interrupt ($\overline{int}$, $\overline{cint}$, or $\overline{smi}$) asserts its request to the core in the SYSA*x* priority position. These bits cannot be changed dynamically. (to change it, software must make sure the corresponding interrupt source is masked or it cannot happen during the change). The definition of SYSA*x*T is as follows: 00 $\overline{int}$ request is asserted to the core for SYSA*x*. 01 $\overline{smi}$ request is asserted to the core for SYSA*x*. 10 $\overline{cint}$ request is asserted to the core for SYSA*x*. 11 Reserved.

17.2.1.10 System External Interrupt Pending Register (IPIC_SEPNR)

Each bit in the System External Interrupt Pending Register (IPIC_SEPNR), shown in Figure 17-13, corresponds to an external interrupt source. When an interrupt is received, the interrupt controller sets the corresponding IPIC_SEPNR bit.

Address: Base + 0x2C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IRQ0[1]	IRQ1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	Note[2]		0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-13. System External Interrupt Pending Register (IPIC_SEPNR)

1 This bit is valid only if the IRQ0 signal is configured as an external maskable interrupt (IPIC_SEMSR[SIRQ0] = 0).

2 These bits reflect the state of external $\overline{\text{IRQ}}$ pins. User should take care to drive all $\overline{\text{IRQ}}$ inputs to an inactive state prior to reset negation.

Table 17-14. IPIC_SEPNR field descriptions

Field	Description
IRQ0	IRQ0 external interrupt source. When an interrupt is received, the interrupt controller sets the corresponding IPIC_SEPNR bit. When a pending interrupt is managed, clear the corresponding IPIC_SEPNR bit. For level triggered case, s/w needs to negate the $\overline{\text{IRQ}}x$ that automatically clears the bit in IPIC_SEPNR, and for edge triggered case, s/w needs to clear IPIC_SEPNR. IPIC_SEPNR bits are cleared by writing ones to them. Because the user can only clear bits in this register, writing zeros to this register has no effect. **Note:** The IPIC_SEPNR bit positions are not changed according to their relative priority.
IRQ1	IRQ1 external interrupt source.

17.2.1.11 System Mixed Interrupt Group A Priority Register (IPIC_SMPRR_A)

The System Mixed Interrupt Group A Priority Register (IPIC_SMPRR_A), shown in Figure 17-14, defines the priority between DIU, DMA2, IRQ0, and IRQ1.

Address: Base + 0x30 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIXA0P				MIXA1P				MIXA2P				MIXA3P			
W													0	0	0	0
Reset	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIXA4P				MIXA5P				MIXA6P				MIXA7P			
W													0	0	0	0
Reset	1	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0

Figure 17-14. System Mixed Interrupt Group A Priority Register (IPIC_SMPRR_A)

Table 17-15. IPIC_SMPRR_A field descriptions

Field	Description
MIXA0P	MIXA0 Priority order. Defines which interrupt source asserts its request in the MIXA0 priority position. The user should not program the same code to more than one priority position (0–7). These bits can be changed dynamically. The definition of MIXA0P is as follows: 000 DIU asserts its request to the MIXA0 position. 001 DMA2 asserts its request to the MIXA0 position. 010 Reserved. 011 Reserved. 100 IRQ0 asserts its request to the MIXA0 position. This field for MIXA0 position is valid (must not be ignored) if IRQ0 signal configured as an external maskable interrupt (IPIC_SEMSR[SIRQ0] = 0). 101 IRQ1 asserts its request to the MIXA0 position. 110 Reserved. 111 Reserved.
MIXA1P–MIXA7P	Same as MIXA0P, but for MIXA1P–MIXA7P.

17.2.1.12 System Mixed Interrupt Group B Priority Register (IPIC_SMPRR_B)

The System Mixed Interrupt Group B Priority Register (IPIC_SMPRR_B), shown in Figure 17-15, defines the priority between PSC0, PSC1, PSC2, and PSC3.

Address: Base + 0x34 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIXB0P				MIXB1P				MIXB2P				MIXB3P			
W													0	0	0	0
Reset	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIXA4P				MIXA5P				MIXA6P				MIXA7P			
W													0	0	0	0
Reset	1	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0

Figure 17-15. System Mixed Interrupt Group B Priority Register (IPIC_SMPRR_B)

Table 17-16. IPIC_SMPRR_B field descriptions

Field	Description
MIXB0P	MIXB0 Priority order. Defines which interrupt source asserts its request in the MIXB0 priority position. The user should not program the same code to more than one priority position (0–7). These bits can be changed dynamically. The definition of MIXB0P is as follows: 000 PSC0 asserts its request to the MIXB0 position. 001 PSC1 asserts its request to the MIXB0 position. 010 PSC2 asserts its request to the MIXB0 position. 011 PSC3 asserts its request to the MIXB0 position. 100 Reserved. 101 Reserved. 110 Reserved. 111 Reserved.
MIXB1P–MIXB7P	Same as MIXB0P, but for MIXB1P–MIXB7P.

17.2.1.13 System External Interrupt Mask Register (IPIC_SEMSR)

Each bit in the System External Interrupt Mask Register (IPIC_SEMSR), shown in Figure 17-16, corresponds to an external interrupt source. Mask an interrupt by clearing the corresponding IPIC_SEMSR bit. An interrupt is unmasked (enabled) by setting the IPIC_SEMSR bit.

When an external interrupt request occurs, the corresponding IPIC_SEPNR bit is set regardless of the IPIC_SEMSR bit. However, if the corresponding IPIC_SEMSR bit is cleared, no interrupt request is passed to the core.

When the IPIC_SEMSR bit is cleared by a user at the same time an interrupt source requests an interrupt service, the request stops. If the user sets the IPIC_SEMSR bit later, a previously pending interrupt request is processed by the core according to its assigned priority. The IPIC_SEMSR can be read at any time.

Address: Base + 0x38 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IRQ0[1]	IRQ1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SIRQ0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-16. System External Interrupt Mask Register (IPIC_SEMSR)

[1] This bit is valid only if the IRQ0 signal is configured as an external maskable interrupt (IPIC_SEMSR[SIRQ0] = 0).

Table 17-17. IPIC_SEMSR field descriptions

Field	Description
IRQ0	IRQ0 external interrupt mask. Mask an interrupt by clearing the IPIC_SEMSR bit. An interrupt can be enabled by setting the corresponding IPIC_SEMSR bit. The IPIC_SEMSR can be read by the user at any time. **Note:** • IPIC_SEMSR bit positions are not affected by their relative priority. • Pending register bits set by multiple interrupt events can be cleared only by clearing all unmasked events in the corresponding event register. • If an IPIC_SEMSR bit is masked at the same time the corresponding IPIC_SEPNR bit causes an interrupt request to the core, the error vector is issued (if no other interrupts pending). Thus, an error vector routine must always be included, even if it contains only an RFI instruction. The error vector cannot be masked.
IRQ1	IRQ1 external interrupt mask.
SIRQ0	Steer IRQ0. 0 IRQ0 is used as external interrupt request. 1 IRQ0 is used as external MCP request.

17.2.1.14 System External Interrupt Control Register (IPIC_SECNR)

The System External Interrupt Control Register (IPIC_SECNR), shown in Figure 17-17, defines the edge detect mode for external $\overline{\text{IRQ}}$ interrupt signals, determines whether the corresponding $\overline{\text{IRQx}}$ signal asserts an interrupt request upon either a high-to-low change or low assertion on the pin. It also defines the IPIC output interrupt type ($\overline{int}$, $\overline{cint}$, or $\overline{smi}$) in the MIXA1–MIXA7 priority positions.

Address: Base + 0x3C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIXB0T		MIXB1T		0	0	0	0	MIXA0T		MIXA1T		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EDI0	EDI1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-17. System External Interrupt Control Register (IPIC_SECNR)

Table 17-18. IPIC_SECNR field descriptions

Field	Description
MIXB0T	MIXB0 priority position IPIC output interrupt Type. Defines which type of the IPIC output interrupt signal (*int*, *cint*, or *smi*) asserts its request to the core in the MIXB0 priority position. These bits can be changed dynamically. The definition of MIXB0T is as follows: 00 $\overline{int}$ request is asserted to the core for MIXB0. 01 $\overline{smi}$ request is asserted to the core for MIXB0. 10 $\overline{cint}$ request is asserted to the core for MIXB0. 11 Reserved.
MIXB1T	Same as MIXB0T, but for MIXB1T.
MIXA0T	MIXA0 priority position IPIC output interrupt Type. Defines which type of the IPIC output interrupt signal (*int*, *cint*, or *smi*) asserts its request to the core in the MIXB0 priority position. These bits can be changed dynamically. The definition of MIXA0T is as follows: 00 $\overline{int}$ request is asserted to the core for MIXA0. 01 $\overline{smi}$ request is asserted to the core for MIXA0. 10 $\overline{cint}$ request is asserted to the core for MIXA0. 11 Reserved.
MIXA1T	Same as MIXA0T, but for MIXA1T.
EDI*x*	Each bit defines the edge detect mode for the external $\overline{IRQ}$ interrupt signals, and determines whether the corresponding $\overline{IRQ}$*x* signal asserts an interrupt request upon either a high-to-low change or low assertion on the pin. The corresponding $\overline{IRQ}$*x* signal asserts an interrupt request as follows: 0 Low assertion on IRQ*x* generates an interrupt request. 1 High-to-low change on IRQ*x* generates an interrupt request.

17.2.1.15 System Error Status Register (IPIC_SERSR)

Each bit in the System Error Status Register (IPIC_SERSR), shown in Figure 17-18, corresponds to an external and an internal non-maskable error source machine check ($\overline{mcp}$). When an error interrupt signal is received, the interrupt controller sets the corresponding IPIC_SERSR bit.

Address: Base + 0x40 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IRQ0	WDT	SBA	0	0	0	0	0	0	TEMP 125C	0	0	0	0	0	0
W	w1c	w1c	w1c							w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-18. System Error Status Register (IPIC_SERSR)

Table 17-19. IPIC_SERSR field descriptions

Field	Description
IRQ0	Each IPIC_SERSR bit corresponds to an external and an internal error source (MCP). When an error interrupt signal is received, the interrupt controller sets the corresponding IPIC_SERSR bit. IPIC_SERSR bits are cleared by writing ones to them. The unmasked event register bits should be cleared before clearing of IPIC_SERSR. Because bits in this register can only be cleared, writing 0s to this register has no effect. IPIC_SERSR bits are reset only by power on reset. Soft and hard reset do not affect IPIC_SERSR bit states.
WDT	
SBA	
TEMP 125C	

17.2.1.16 System Error Mask Register (IPIC_SERMR)

Each bit in the System Error Mask Register (IPIC_SERMR), shown in Figure 17-19, corresponds to an external and an internal *mcp* source. The user masks an MCP by clearing and enables an interrupt by setting the corresponding IPIC_SERMR bit. When a masked MCP occurs, the corresponding IPIC_SERSR bit is set regardless of the IPIC_SERMR bit, although no MCP request is passed to the core. The IPIC_SERMR can be read by the user at any time.

Address: Base + 0x44 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IRQ0	WDT	SBA	0	0	0	0	0	1	TEMP 125C	0	0	0	0	0	0
W																
Reset	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-19. System Error Mask Register (IPIC_SERMR)

Table 17-20. IPIC_SERMR field descriptions

Field	Description
IRQ0	Each bit in the System Error Status Register (IPIC_SERMR), shown in Figure 17-19, corresponds to an external and an internal MCP source. Mask an MCP by clearing and enables an interrupt by setting the corresponding IPIC_SERMR bit. When a masked MCP occurs, the corresponding IPIC_SERSR bit is set, regardless of the IPIC_SERMR bit, although no MCP request is passed to the core. The IPIC_SERMR can be read by the user at any time.
WDT	
SBA	
TEMP 125C	

17.2.1.17 System External Interrupt Polarity Check Register (IPIC_SEPCR)

Each bit in the System External Interrupt Polarity Check Register (IPIC_SEPCR), shown in Figure 17-20, defines the polarity for each one of the external IRQ*n* interrupt signals and determines whether the corresponding IRQn signal is treated as active low or active high signal. The active low signals will assert an interrupt request upon either a high-to-low change or assertion (low state) on the pin. The active high signals will assert an interrupt request upon either a low-to-high change or assertion (high state) on the pin.

Address: Base + 0x4C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EIP0	EIP1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-20. System External Interrupt Polarity Check Register (IPIC_SEPCR)

Table 17-21. IPIC_SEPCR field descriptions

Field	Description
EIP0	External Interrupt 0 Polarity. Defines the active state for the external $\overline{\text{IRQ}}$*n* interrupt signals. 0 Active Low. 1 Active High.
EIP1	External Interrupt 1 Polarity. Defines the active state for the external $\overline{\text{IRQ}}$*n* interrupt signals. 0 Active Low. 1 Active High.

17.2.1.18 System Internal Interrupt Force Registers High and Low (IPIC_SIFCR_H and IPIC_SIFCR_L)

Each bit in the System Internal Interrupt Force Registers High and Low (IPIC_SIFCR_H and IPIC_SIFCR_L), shown in Figure 17-21 and Figure 17-22, corresponds to an internal interrupt source. When a bit is set, the interrupt controller generates the corresponding interrupt (sets the corresponding IPIC_SIPNR*x* bit).

The IPIC_SIFCR*x* registers can be read by the user at any time.

Address: Base + 0x50 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PSC4	PSC5	PSC6	PSC7	PSC8	PSC9	GPT8	GPT9	FIFOC	0	0	USB2 OTG1	USB2 OTG2	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPT10	GPT11	SDHC2	FEC1	FEC2	NFC	LPC	SDHC1	I2C1	I2C2	I2C3	MSCAN1	MSCAN2	BDLC	GPT0	GPT1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-21. System Internal Interrupt Force Registers High (IPIC_SIFCR_H)

Table 17-22. IPIC_SIFCR_H field descriptions

Field	Description
PSC4	Each bit corresponds to an interrupt source. Force an interrupt by setting the IPIC_SIFCR*x* bit. An interrupt can be enabled by setting the corresponding IPIC_SIFCR*x* bit. **Note:** IPIC_SIFCR*x* bit positions are not changed according to their relative priority.
PSC5	
PSC6	
PSC7	
PSC8	
PSC9	
GPT8	
GPT9	
FIFOC	
USB2OTG1	
USB2OTG2	
GPT10	
GPT11	
SDHC2	
FEC1	
FEC2	
NFC	
LPC	
SDHC1	
I2C1	
I2C2	
I2C3	
MSCAN1	

Table 17-22. IPIC_SIFCR_H field descriptions (continued)

Field	Description
MSCAN2	
BDLC	
GPT0	
GPT1	

Address: Base + 0x54 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DIU	DMA2	0	0	PSC0	PSC1	PSC2	PSC3	GPT2	GPT3	GPT4	GPT5	GPT6	GPT7	GPIO 1	RTC SEC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RTC ALARM	DDR	SBA	PMC	USB2 OTG1 WKUP	USB2 OTG2 WKUP	GPIO2	TEMP 105C	IIM	PRI-OMON	MSCAN3	MSCAN4	GPT12	GPT13	GPT14	GPT15
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-22. System Internal Interrupt Force Registers Low (IPIC_SIFCR_L)

Table 17-23. IPIC_SIFCR_L field descriptions

Field	Description
DIU	Each bit corresponds to an interrupt source. Force an interrupt by setting the IPIC_SIFCR*x* bit. An interrupt can be enabled by setting the corresponding IPIC_SIFCR*x* bit. **Note:** IPIC_SIFCR*x* bit positions are not changed according to their relative priority.
DMA2	
PSC0	
PSC1	
PSC2	
PSC3	
GPT2	
GPT3	
GPT4	
GPT5	
GPT6	
GPT7	
GPIO1	
RTC SEC	
RTC ALARM	
DDR	

Table 17-23. IPIC_SIFCR_L field descriptions (continued)

Field	Description
SBA	
PMC	
USB2OTG1 WKUP	
USB2OTG2 WKUP	
GPIO2	
TEMP 105C	
IIM	
PRIOMON	
MSCAN3	
MSCAN4	
GPT12	
GPT13	
GPT14	
GPT15	

17.2.1.19 System External Interrupt Force Register (IPIC_SEFCR)

Each bit in the System External Interrupt Force Register (IPIC_SEFCR), shown in Figure 17-17, corresponds to an external interrupt source. When a bit is set, the interrupt controller generates the corresponding external interrupt (sets the corresponding IPIC_SEPNR bit).

The IPIC_SEFCR can be read by the user at any time.

Address: Base + 0x58 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IRQ0[1]	IRQ1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-23. System External Interrupt Force Register (IPIC_SEFCR)

[1] This bit is valid only if the IRQ0 signal is configured as an external maskable interrupt (IPIC_SEMSR[SIRQ0] = 0)

Table 17-24. IPIC_SEFCR field descriptions

Field	Description
IRQ0	Each bit corresponds to an external interrupt source. Force an interrupt by setting the IPIC_SIFCR*x* bit. An interrupt can be enabled by setting the corresponding IPIC_SIFCR*x* bit. **Note:** IPIC_SIFCR*x* bit positions are not changed according to their relative priority.
IRQ1	

17.2.1.20 System Error Force Register (IPIC_SERFR)

Each bit in the System Error Force Register (IPIC_SERFR), shown in Figure 17-24, corresponds to an external MCP source. When a bit is set, the interrupt controller generates the corresponding MCP interrupt (sets the corresponding IPIC_SERSR bit).

The IPIC_SERFR can be read by the user at any time.

Address: Base + 0x5C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IRQ0[1]	WDT	SBA	0	0	0	0	0	TEST	TEMP 125C	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-24. System Error Force Register (IPIC_SERFR)

1 1. This bit is valid only if the IRQ0 signal is configured as an external MCP interrupt (IPIC_SEMSR[SIRQ0] = 1)

Table 17-25. IPIC_SERFR field descriptions

Field	Description
IRQ0	Each bit corresponds to an external MCP source. The user can force an MCP by setting the IPIC_SERFR bit. **Note:** IPIC_SERFR bit positions are not affected by their relative priority.
WDT	
SBA	
TEST	
TEMP 125C	

17.2.1.21 System Critical Interrupt Vector Register (IPIC_SCVCR)

The System Critical Interrupt Vector Register (IPIC_SCVCR), shown in Figure 17-25, contains a 7-bit code (Table 17-26) representing the unmasked critical interrupt ($\overline{cint}$) source of the highest priority level.

Address: Base + 0x60 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	CVEC						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-25. System Critical Interrupt Vector Register (IPIC_SCVCR)

Table 17-26. IPIC_SCVCR field descriptions

Field	Description
CVEC	Critical interrupt vector. Specifies a 7-bit unique number of the IPIC's highest priority critical interrupt source, pending to the core. When a critical interrupt request occurs, IPIC_SCVCR can be read. If there are multiple critical interrupt sources, IPIC_SCVCR latches the highest priority critical interrupt. The CVEC field correctly reflects all of the interrupt vectors (See Table 17-4 for details). The value of SCVEC cannot change while it is being read. **Note:** While the upper 24 bits of this register are set to zero at the release of PORESET, they can and will change during device operation. Therefore, when using the value of the CVEC field, be sure to mask off the upper 24 bits of this register.

17.2.1.22 System Management Interrupt Vector Register (IPIC_SMVCR)

The System Management Interrupt Vector Register (IPIC_SMVCR), shown in Figure 17-26, contains a 7-bit code (Table 17-27) representing the unmasked system management interrupt ($\overline{\text{SMI}}$) source of the highest priority level.

Address: Base + 0x64 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	MVEC						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-26. System Management Interrupt Vector Register (IPIC_SMVCR)

Table 17-27. IPIC_SMVCR field descriptions

Field	Description
MVEC	System management interrupt vector. Specifies a 7-bit unique number of the IPIC's highest priority system management interrupt source, pending to the core. When a system management interrupt request occurs, IPIC_SMVCR can be read. If there are multiple system management interrupt sources, IPIC_SMVCR latches the highest priority system management interrupt. The MVEC field correctly reflects all interrupt vectors (See Table 17-4 for details). The value of SMVEC cannot change while it is being read. **Note:** While the upper 24 bits of this register are set to zero at the release of PORESET, they can and will change during device operation. Therefore, when using the value of the MVEC field, be sure to mask off the upper 24 bits of this register.

17.3 Functional Description

The following sections describe the type of interrupts, interrupt configurations, and their priorities.

17.3.1 Interrupt Types

The IPIC is responsible for receiving hardware-generated interrupts from different sources (both internal and external) along with prioritizing and delivering them to the CPU for servicing. The interrupt sources are controlled by the IPIC unit and may cause three types of exceptions in the processor core. The $\overline{int}$ signal is the main interrupt output from the IPIC to the processor core and causes the external interrupt exception.The $\overline{cint}$ signal is the critical interrupt output from the IPIC to the processor core and causes the critical external interrupt exception. The $\overline{smi}$ signal is the system management interrupt output from the IPIC to the processor core and causes the system management interrupt exception. The machine check exception is caused by the internal $\overline{mcp}$ signal generated by the IPIC, informing the processor of error conditions, assertion of the external MCP request, and other conditions.

17.3.2 Interrupt Configuration

Figure 17-28 shows the interrupt configuration of the MPC5125.

The interrupt controller allows masking of each interrupt source. When an unmasked interrupt source is pending in the IPIC_SIPNR*x* register, the interrupt controller sends an interrupt request to the core. When an interrupt is taken, the interrupt mask bit in the machine state register is cleared to disable further interrupt requests to the Power Architecture core until software can manage them.

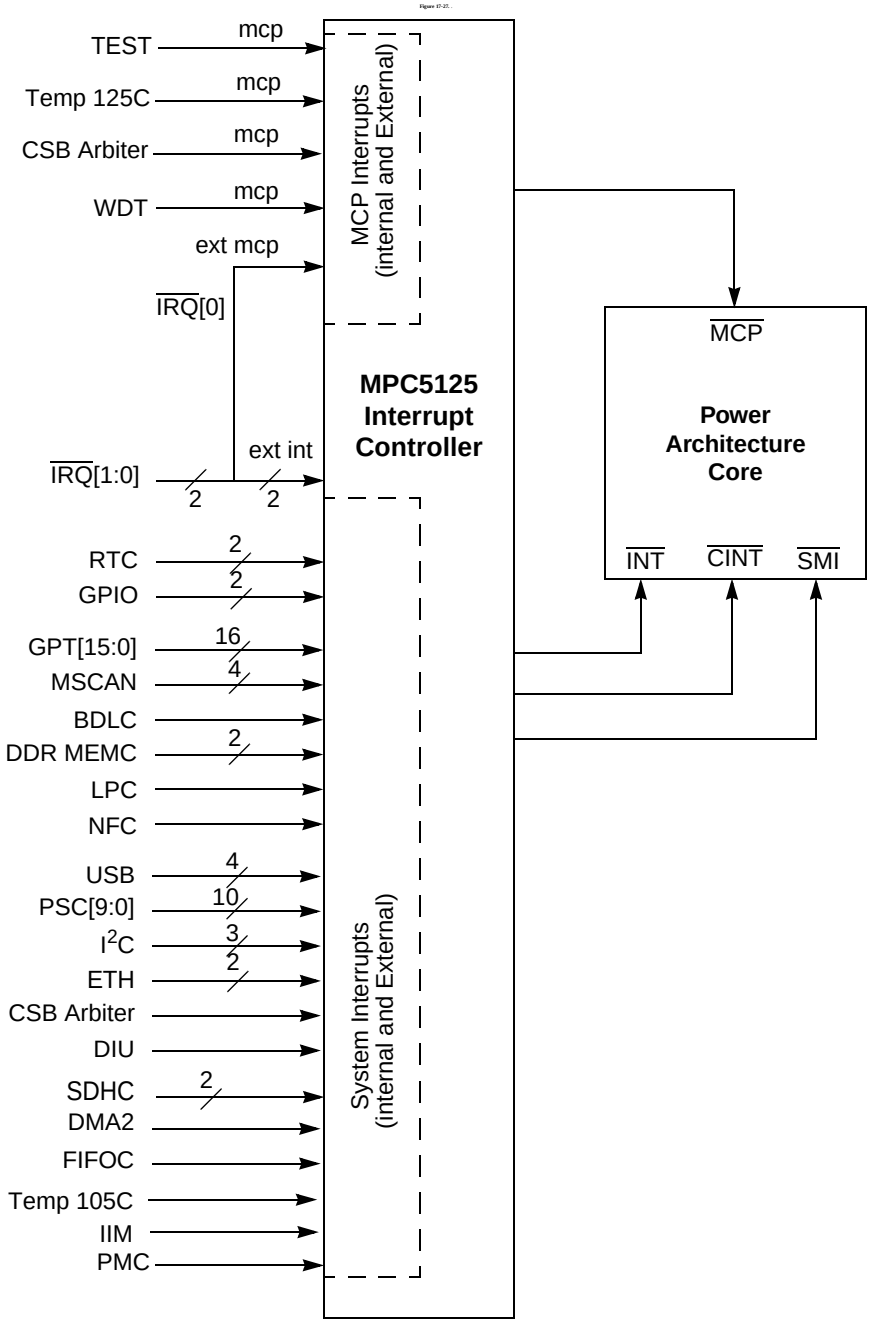
All interrupt sources are prioritized and bits are set in the system interrupt pending register (IPIC_SIPNR*x*, IPIC_SEPNR) as interrupts occur regardless of whether they are masked in the IPIC. The prioritization of the interrupt sources is flexible within the following groups:

- The relative priority of the PSC4, PSC5, PSC6, PSC7, PSC8, PSC9, GPT8, and GPT9 internal interrupt signals can be modified.
- The relative priority of the FIFOCUSB2OTG1, and USB2OTG2 internal interrupt signals can be modified.

- The relative priority of the GPT10, GPT11, SDHC2, FEC1, FEC2, NFC, LPC, and SDHC1 internal interrupt signals can be modified.
- The relative priority of the I2C1, I2C2, I2C3, MSCAN1, MSCAN2, BDLC, GPT0, and GPT1 internal interrupt signals can be modified.
- The relative priority of the IRQ0, IRQ1 external interrupts, and DIU, DMA2 internal interrupts can be modified.
- The relative priority of the PSC0, PSC1, PSC2 and PSC3 internal interrupts can be modified.
- One interrupt source can be assigned to be the programmable highest priority.

All other interrupt sources have a fixed interrupt priority. For details see Table 17-28.

The SIVEC is updated with a 7-bit vector corresponding to the sub-block with the highest current priority.



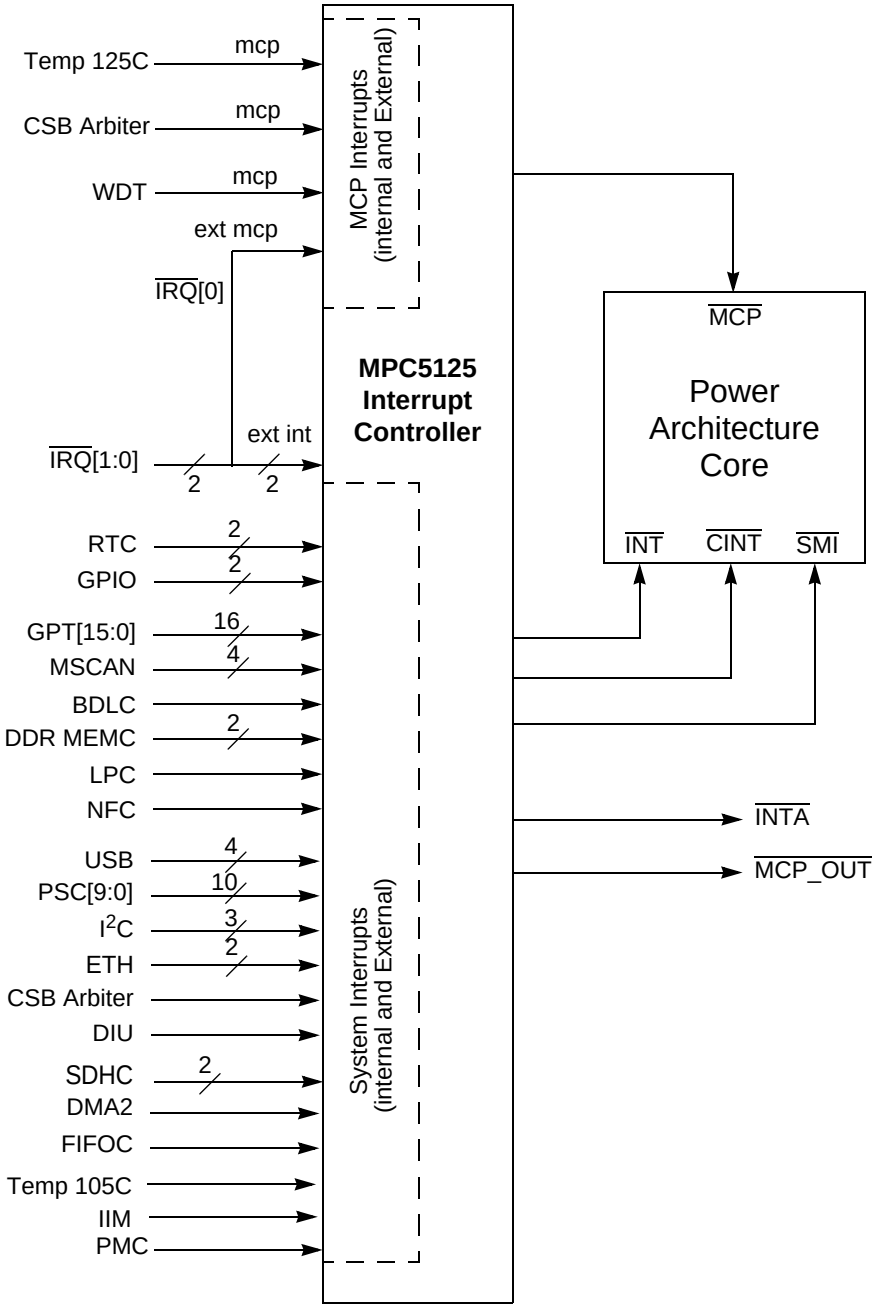


Figure 17-28. MPC5125 Interrupt Structure

17.3.3 Internal Interrupts Group Relative Priority

The relative priority in each internal group is programmable and can be changed dynamically. The group priorities are programmed in the IPIC internal interrupt priority registers (SIPRR*x*) and can be changed dynamically to implement a rotating priority.

In addition, the grouping of the locations of the interrupt entries has the following two options:

- Grouped. In the group scheme, all interrupts are grouped together at the top of Table 17-28, ahead of most other interrupt sources. This scheme is ideal for applications where all interrupt sources function at a very high data rate and interrupt latency is very important.
- Spread. In the spread scheme, priorities are spread over Table 17-28 so other sources can have lower interrupt latencies. This scheme is also programmed but cannot be changed dynamically.

17.3.4 Mixed Interrupts Group Relative Priority

The relative priority between as many as four internal and four external interrupts in each group is programmable and can be changed dynamically. The group priorities are programmed in the IPIC mixed interrupt priority registers (SMPRR*x*) and can be changed dynamically to implement a rotating priority.

In addition, the grouping of the locations of the mixed interrupt entries has the following two options:

- Grouped. In the group scheme, all interrupts are grouped together at the top of the priority table, ahead of most other interrupt sources. See Table 17-28 for more information. This scheme is ideal for applications where all interrupt sources function at a very high data rate and interrupt latency is very important.
- Spread. In the spread scheme, priorities are spread over the table so other sources can have lower interrupt latencies. This scheme is also programmed but cannot be changed dynamically.

17.3.5 Highest Priority Interrupt

In addition to the group relative priority option, IPIC_SICFR[HPI] can be used to specify one interrupt source as having the highest priority. This interrupt remains within the same interrupt level as the other interrupt controller interrupts, but is serviced before any other interrupt in Table 17-28.

If the highest priority feature is not used, the IPIC selects the interrupt request in MIXA0 to be the highest priority interrupt and the standard interrupt priority order is used from Table 17-28. IPIC_SICFR[HPI] can be updated dynamically to allow changing a normally low priority source into a high priority-source for a period as needed.

17.3.6 Interrupt Source Priorities

Each of the IPIC's internal and external interrupt sources can independently assert one interrupt request to the core. Table 17-28 shows the prioritization of these interrupt sources. As described in previous sections, flexibility exists in the relative ordering of the interrupts, but in general, relative priorities are as shown. A single interrupt priority number is associated with each table entry.

Table 17-28. Interrupt Source Priority Levels

Priority Level	Interrupt Source Description	Multiple Events
0	Highest	—
1	MIXA0 (Spread)	Yes (No for ext. interrupts)
2	MIXA0 (Grouped)	Yes (No for ext. interrupts)
3	MIXA1 (Grouped)	Yes (No for ext. interrupts)

Table 17-28. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
4	MIXA2 (Grouped)	Yes (No for ext. interrupts)
5	MIXA3 (Grouped)	Yes (No for ext. interrupts)
6	MIXB0 (Spread)	Yes (No for ext. interrupts)
7	SYSB0 (Grouped)	Yes
8	SYSB1 (Grouped)	Yes
9	SYSB2 (Grouped)	Yes
10	SYSB3 (Grouped)	Yes
11	MIXA1 (Spread)	Yes (No for ext. interrupts)
12	SYSB4 (Grouped)	Yes
13	SYSB5 (Grouped)	Yes
14	SYSB6 (Grouped)	Yes
15	SYSB7 (Grouped)	Yes
16	MIXB0 (Grouped)	Yes (No for ext. interrupts)
17	MIXB1 (Grouped)	Yes (No for ext. interrupts)
18	MIXB2 (Grouped)	Yes (No for ext. interrupts)
19	MIXB3 (Grouped)	Yes (No for ext. interrupts)
20	MIXB1 (Spread)	Yes (No for ext. interrupts)
21	SYSA0 (Grouped)	Yes
22	SYSA1 (Grouped)	Yes
23	SYSA2 (Grouped)	Yes
24	SYSA3 (Grouped)	Yes
25	MIXA2 (Spread)	Yes (No for ext. interrupts)
26	SYSA4 (Grouped)	Yes
27	SYSA5 (Grouped)	Yes
28	SYSA6 (Grouped)	Yes
29	SYSA7 (Grouped)	Yes
30	MIXA4 (Grouped)	Yes (No for ext. interrupts)
31	MIXA5 (Grouped)	Yes (No for ext. interrupts)
32	MIXA6 (Grouped)	Yes (No for ext. interrupts)
33	MIXA7 (Grouped)	Yes (No for ext. interrupts)
34	MIXB2 (Spread)	Yes (No for ext. interrupts)
35	SYSC0 (Grouped)	Yes
36	SYSC1 (Grouped)	Yes
37	SYSC2 (Grouped)	Yes
38	SYSC3 (Grouped)	Yes

Table 17-28. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
39	MIXA3 (Spread)	Yes (No for ext. interrupts)
40	SYSC4 (Grouped)	Yes
41	SYSC5 (Grouped)	Yes
42	SYSC6 (Grouped)	Yes
43	SYSC7 (Grouped)	Yes
44	MIXB4 (Grouped)	Yes (No for ext. interrupts)
45	MIXB5 (Grouped)	Yes (No for ext. interrupts)
46	MIXB6 (Grouped)	Yes (No for ext. interrupts)
47	MIXB7 (Grouped)	Yes (No for ext. interrupts)
48	MIXB3 (Spread)	Yes (No for ext. interrupts)
49	SYSD0 (Grouped)	Yes
50	SYSD1 (Grouped)	Yes
51	SYSD2 (Grouped)	Yes
52	SYSD3 (Grouped)	Yes
53	MIXA4 (Spread)	Yes (No for ext. interrupts)
54	SYSD4 (Grouped)	Yes
55	SYSD5 (Grouped)	Yes
56	SYSD6 (Grouped)	Yes
57	SYSD7 (Grouped)	Yes
58	MIXB4 (Spread)	Yes (No for ext. interrupts)
59	GPT2	Yes
60	SYSB0 (Spread)	Yes
61	SYSA0 (Spread)	Yes
62	GPT3	Yes
63	SYSC0 (Spread)	Yes
64	SYSD0 (Spread)	Yes
65	Reserved	No
66	GPT4	Yes
67	MIXA5 (Spread)	Yes (No for ext. interrupts)
68	GPT5	Yes
69	SYSB1 (Spread)	Yes
70	SYSA1 (Spread)	Yes
71	GPT6	Yes
72	SYSC1 (Spread)	Yes
73	SYSD1 (Spread)	Yes

Table 17-28. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
74	Reserved	No
75	GPT7	Yes
76	MIXB5 (Spread)	Yes (No for ext. interrupts)
77	GPIO1	Yes
78	SYSB2 (Spread)	Yes
79	SYSA2 (Spread)	Yes
80	RTC SEC	Yes
81	SYSC2 (Spread)	Yes
82	SYSD2 (Spread)	Yes
83	Reserved	No
84	RTC ALARM	Yes
85	MIXA6 (Spread)	Yes (No for ext. interrupts)
86	DDR	Yes
87	SYSB3 (Spread)	Yes
88	SYSA3 (Spread)	Yes
89	SBA	Yes
90	SYSC3 (Spread)	Yes
91	SYSD3 (Spread)	Yes
92	Reserved	No
93	PMC	Yes
94	MIXB6 (Spread)	Yes (No for ext. interrupts)
95	USB2OTG1 WKUP	Yes
96	SYSB4 (Spread)	Yes
97	SYSA4 (Spread)	Yes
98	USB2OTG2 WKUP	Yes
99	SYSC4 (Spread)	Yes
100	SYSD4 (Spread)	Yes
101	Reserved	No
102	GPIO2	Yes
103	MIXA7 (Spread)	Yes (No for ext. interrupts)
104	TEMP 105C	Yes
105	SYSB5 (Spread)	Yes
106	SYSA5 (Spread)	Yes

Table 17-28. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
107	IIM	Yes
108	SYSC5 (Spread)	Yes
109	SYSD5 (Spread)	Yes
110	Reserved	No
111	PRIOMON	Yes
112	MIXB7 (Spread)	Yes (No for ext. interrupts)
113	MSCAN3	Yes
114	SYSB6 (Spread)	Yes
115	SYSA6 (Spread)	Yes
116	MSCAN4	Yes
117	SYSC6 (Spread)	Yes
118	SYSD6 (Spread)	Yes
119	Reserved	No
120	GPT12	Yes
121	GPT13	Yes
122	SYSB7 (Spread)	Yes
123	SYSA7 (Spread)	Yes
124	GPT14	Yes
125	SYSC7 (Spread)	Yes
126	SYSD7 (Spread)	Yes
127	Reserved	No
128	GPT15	Yes

17.3.7 Masking Interrupt Sources

By programming the system interrupt mask registers, IPIC_SIMSR*x* and IPIC_SEMSR, the user can mask interrupt requests to the core. Each IPIC_SIMSR*x* and IPIC_SEMSR bit corresponds to an interrupt source. To enable an interrupt, set the corresponding IPIC_SIMSR or IPIC_SEMSR bit. When a masked interrupt source has a pending interrupt request, the corresponding IPIC_SIPNR*x* or IPIC_SEMSR bit is set, even though the interrupt is not generated to the core. The user can mask all interrupt sources to implement a polling interrupt servicing scheme.

When an interrupt source has multiple interrupting events, the user can individually mask these events by programming a mask register within that particular block. Table 17-28 shows which interrupt sources have multiple interrupting events. Figure 17-29 shows an example of how the masking occurs, using a DDR as an example.

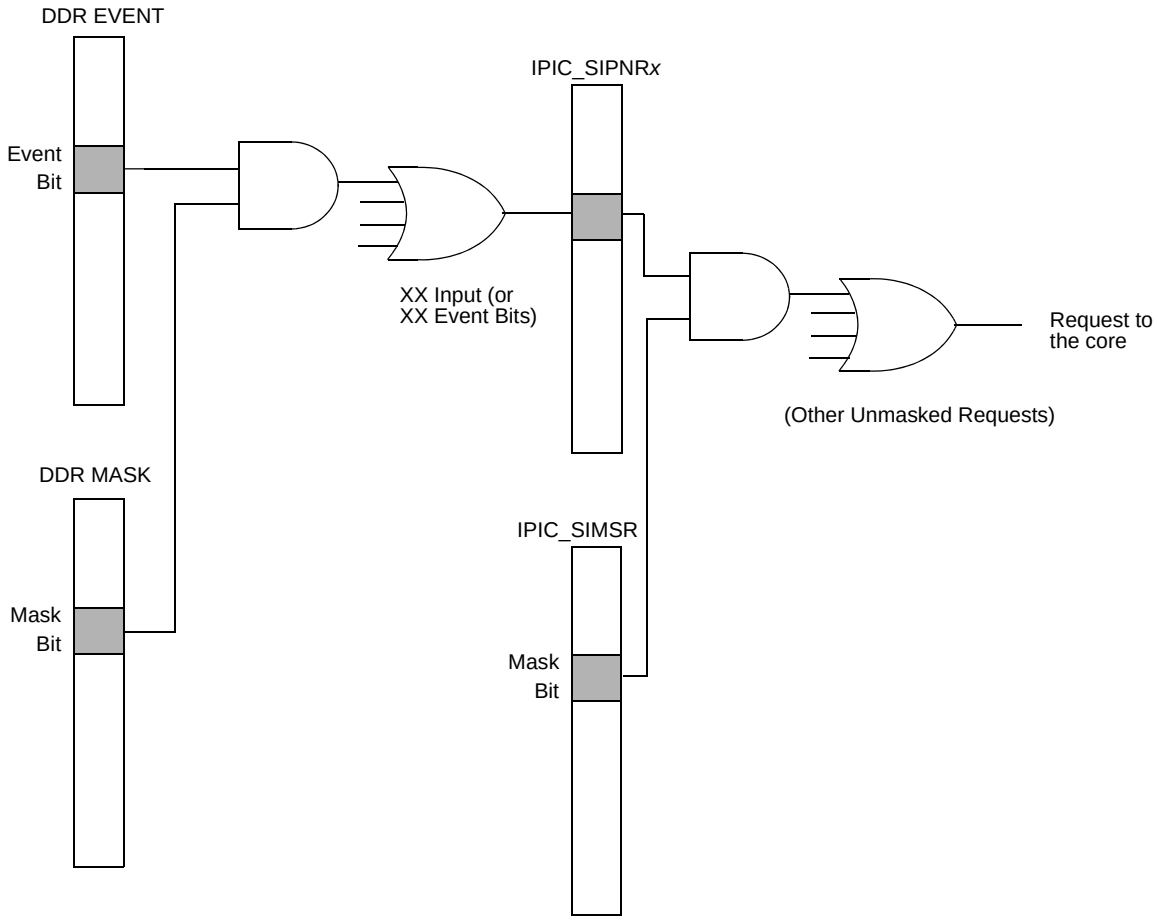


Figure 17-29. DDR Interrupt Request Masking

17.3.8 Interrupt Vector Generation and Calculation

Pending unmasked interrupts are presented to the core in order of priority according to Table 17-28. The interrupt vector that allows the core to locate the interrupt service routine is made available to the core by interrupt handler software reading IPIC_SIVCR, IPIC_SCVCR or IPIC_SMVCR. The interrupt controller passes an interrupt vector corresponding to the highest-priority, unmasked, pending interrupt in response to a read of IPIC_SIVCR, IPIC_SCVCR or IPIC_SMVCR. Table 17-4 lists the encodings for the seven low-order bits of the interrupt vector.

17.3.9 Machine Check Interrupts

There are 5 non-maskable machine check interrupts (MCP), coming from the internal sources and one programmable MCP from the external source.

Chapter 18
IIM/Fusebox

18.1 Introduction

The IC Identification Module (IIM) provides an interface for reading and programming information stored in on-chip fuse elements.

18.2 Overview

The IIM provides the primary user-visible mechanism for interfacing with on-chip fuse elements. Among other uses, e-fuses can be used for unique chip identifiers, cryptographic keys, and various control signals requiring permanent non-volatility.

The IIM consists of a master controller, a fuse value shadow cache, and a set of registers to hold the values of signals visible outside the module. Two 256-bit fuse banks are implemented on the MPC5125.

The e-Fuses may be blown under software control at the customer factory or in the field. They include a mechanism to inhibit further blowing of fuses (write-protect) to support secure computing environments. The fuse values may also be overridden by software without modifying the fuse element. Similar to the write-protect functionality, the override functionality can also be permanently disabled.

18.2.1 Features

- Two fuse banks, each fuse bank size is 256 bits
- Ability to write-protect e-Fuses on a per-bank basis

18.2.2 Modes of Operation

The IIM is in its functional mode (all specified functionality available) any time it is out of reset and supplied with the proper clocks.

For programming the external FUSE programming supply, AVDD_FUSEWR must be applied.

18.3 Memory Map and Register Definition

Section 18.3.2, “Register Descriptions,” provides the detailed register descriptions for all of the IIM registers

18.3.1 Memory Map

All registers are 8-bit wide, but addressable on 32-bit boundaries. Only the least significant 8 bits (the usable bits) of each register are shown in the following diagrams. The top most significant bits always read as 0 and writes to them are ignored. Table 18-1 shows the IIM memory map.

Table 18-1. IIM memory map

Offset from IIM_BASE (0xFF40_B000)[1]	Register	Access[2]	Reset Value[3]	Section/Page
0x000	IIM_STAT—Status register	R/W	0x0000_0000	18.3.2.1/18-458
0x004	IIM_STATM—Status IRQ Mask register	R/W	0x0000_0000	18.3.2.2/18-459
0x008	IIM_ERR—Module Errors register	R/W	0x0000_000U	18.3.2.3/18-460
0x00C	IIM_EMASK—Error IRQ Mask register	R/W	0x0000_0000	18.3.2.4/18-461
0x010	IIM_FCTL—Fuse Control register	R/W	0x0000_0030	18.3.2.5/18-462
0x014	IIM_UA—Upper Address register	R/W	0x0000_0000	18.3.2.6/18-463
0x018	IIM_LA—Lower Address register	R/W	0x0000_0000	18.3.2.7/18-464
0x01C	IIM_SDAT—Explicit Sense Data register	R	0x0000_0000	18.3.2.8/18-465
0x020–0x027	Reserved			
0x028	IIM_PREG_P—Program Protection register	R/W	0x0000_0000	18.3.2.9/18-465
0x02C–0x03B	Reserved			
0x03C	IIM_DIVIDE—Divide Factor register	R/W	0x0000_0042	18.3.2.10/18-466
0x040–0xBFF	Reserved			
0xC00	IIM_FBAC1—Fuse Bank 1 Protection Register	R/W	0x0000_00UU	18.3.2.11/18-467
0xC04–0xC7F	IIM_FB1W1—Fuse Bank 1 Data (available for user)	R/W	0x0000_00UU	18.3.2.12/18-468
0xC80–0xFFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

3 In this column, the symbol “U” indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

18.3.2 Register Descriptions

This section contains detailed descriptions of the IIM registers.

18.3.2.1 Status (IIM_STAT) Register

Figure 18-1 shows the bits in the Status (IIM_STAT) register. Table 18-2 describes the bit fields.

Address: Base + 0x000 Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	BUSY	0	0	0	0	0	PRGD	SNSD
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	—	—	—	—	—	0

Figure 18-1. Status (IIM_STAT) Register

Table 18-2. IIM_STAT field descriptions

Field	Description
BUSY	Indicates whether the IIM is busy with a program or sense cycle. Any attempt to access the IIM registers other than IIM_STAT while it is busy with a program or sense cycle (BUSY asserted) results in a bus error. 0 The IIM is not busy with a program or sense cycle. 1 The IIM is busy with a program or sense cycle.
PRGD	Program Done. Indicates an e-Fuse program operation is done. Assertion causes an interrupt request if PRGD_M is set in the status IRQ mask register. This bit is automatically set by hardware upon completion of an e-Fuse program cycle; software must clear the bit by writing 1 to it. 0 Program operation has not finished (read); no meaning (write). 1 Program operation has finished (read); clear bit (write).
SNSD	Explicit Sense Cycle Done. Indicates that an explicit fuse sense cycle is done, and the data is available in the IIM_SDAT register. Assertion causes an interrupt request if SNSD_M is set in the status IRQ mask register. This bit is automatically set by hardware and must be cleared by software by writing 1 to it. 0 No explicit sense cycle has finished (read); no meaning (write). 1 An explicit sense cycle has finished (read); clear bit (write).

18.3.2.2 Status IRQ Mask (IIM_STATM) Register

Figure 18-2 shows the bits in the Status IRQ Mask (IIM_STATM) register. Table 18-3 describes the bit fields.

Address: Base + 0x004 Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PRGD_M	SNSD_M
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-2. Status IRQ Mask (IIM_STATM) Register

Table 18-3. IIM_STATM field descriptions

Field	Description
PRGD_M	Program Mask. Masks or unmasks IRQ generation due to PRGD events. 0 PRGD events do not cause an IRQ. 1 PRGD events cause an IRQ.
SNSD_M	Explicitly Sense Cycle Done Mask. Masks or unmasks IRQ generation due to SNSD events. 0 SNSD events do not cause an IRQ. 1 SNSD events cause an IRQ.

18.3.2.3 Module Errors (IIM_ERR) Register

Figure 18-3 shows the bits in the Module Errors (IIM_ERR) register. Table 18-4 describes the bit fields.

Address: Base + 0x008 Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	WPE	OPE	RPE	0	SNSE	PARI-TYE	0
W										w1c	w1c	w1c		w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	—	u

Figure 18-3. Module Errors (IIM_ERR) Register

Table 18-4. IIM_ERR field descriptions

Field	Description
WPE	Write Protect Error. Indicates an e-Fuse program operation was attempted to a write-protected fuse bank, a locked words, or when the value of IIM_PREG_P is not 0xAA. Assertion causes an interrupt request if WPE_M is set in the errors IRQ mask register. This bit is automatically set by hardware and must be cleared by software by writing 1 to it. 0 There was no write-protect error (read); no meaning (write). 1 There was a write-protect error (read); clear bit (write).
OPE	Override Protect Error. Indicates an attempt was made to override the values in an override-protected fuse bank or a locked words. Assertion causes an interrupt request if OPE_M is set in the errors IRQ mask register. This bit is automatically set by hardware and must be cleared by software by writing 1 to it. 0 There was no override-protect error (read); no meaning (write). 1 There was an override-protect error (read); clear bit (write).
RPE	Read Protect Error. Indicates an attempt was made to read values from a read-protected fuse bank or SCC. Assertion causes an interrupt request if RPE_M is set in the errors IRQ mask register. This bit is automatically set by hardware and must be cleared by software by writing 1 to it. 0 There was no read-protect error (read); no meaning (write). 1 There was a read-protect error (read); clear bit (write).
SNSE	Explicit Sense Cycle Error. Indicates that an explicit fuse sense was refused, because FBESP is set to 1 or more than two bits of SNS_N, SNS_1, SNS_0, and PRG are asserted at the same moment. Assertion causes an interrupt request if SNSE_M is set in the errors IRQ mask register. This bit is automatically set by hardware and must be cleared by software by writing 1 to it. 0 There was no explicit sense error (read); no meaning (write). 1 There was an explicit sense error (read); clear bit (write).
PARITYE	Parity Error of Cache. Indicates that a parity error was detected in the hardware or software fuse cache. Assertion causes an interrupt request if PARITYE_M is set in the errors IRQ mask register. This bit is automatically set by hardware and must be cleared by software by writing 1 to it. Parity is calculated for each fuse word when it is written into hardware cache for data correction check. Odd parity is used across data bits. That is, if the data byte has odd number of 1, the parity bit is set to 1. When fuse data is read, parity is calculated again. If the two parity valus are mismatched, a parity error is reported 0 There was no parity error (read); no meaning (write). 1 There was a parity error (read); clear bit (write).

18.3.2.4 Error IRQ Mask (IIM_EMASK) Register

Figure 18-4 shows the bits in the Error IRQ Mask (IIM_EMASK) register. Table 18-5 describes the bit fields.

Address: Base + 0x00C Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	WPE_M	OPE_M	RPE_M	0	SNSE_M	PARITYE_M	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-4. Error IRQ Mask Register (IIM_EMASK)

Table 18-5. IIM_EMASK field descriptions

Field	Description
WPE_M	Write Protect Error Mask. Masks or unmasks IRQ generation due to WPE events. 0 WPE events do not cause an IRQ. 1 WPE events cause an IRQ.
OPE_M	Override Protect Error Mask. Masks or unmasks IRQ generation due to OPE events. 0 OPE events do not cause an IRQ. 1 OPE events cause an IRQ.
RPE_M	Read Protect Error Mask. Masks or unmasks IRQ generation due to RPE events. 0 RPE events do not cause an IRQ. 1 RPE events cause an IRQ.
SNSE_M	Explicit Sense Cycle Error Mask. Masks or unmasks IRQ generation due to SNSE events. 0 SNSE events do not cause an IRQ. 1 SNSE events cause an IRQ.
PARITYE_M	Parity Error of Cache Mask. Masks or unmasks IRQ generation due to PARITYE events. 0 PARITYE events do not cause an IRQ. 1 PARITYE events cause an IRQ.

18.3.2.5 Fuse Control (IIM_FCTL) Register

Figure 18-5 shows the bits in the Fuse Control (IIM_FCTL) register. Table 18-6 describes the bit fields.

Address: Base + 0x010 Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	PRG_LENGTH[2:0]			ESNS_N	ESNS_0	ESNS_1	PRG
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Figure 18-5. Fuse Control (IIM_FCTL) Register

Table 18-6. IIM_FCTL field descriptions

Field	Description
PRG_LENGTH [2:0]	Program Length. These thee bits define the length of the program pulse. PRG_LENGTH × (period of 32 kHz clock).
ESNS_N	Explicit Sense—Normal. Writing 1 to this bit initiates an unstressed (normal) explicit sense cycle. Reading this bit always returns 0. FSM generates a done signal when the operation completes. This bit is cleared automatically by hardware when sense operation completes. Only one of ESNS_N, ESNS_0, ESNS_1, and PRG can be asserted. Otherwise, ESNS_E is asserted to indicate this error. 0 Return 0 for all read (read); No meaning (write). 1 Initiate an unstressed explicit sense cycle (write).
ESNS_0	Explicit Sense—0 Stressed. Writing 1 to this bit initiates a 0-stressed explicit sense cycle. Reading this bit always returns 0. FSM generates a done signal when the operation completes. This bit is cleared automatically by hardware when sense operation completes. During 0-stressed explicit sense cycles, the EPM_READSENSE0 signal is asserted to the fuse banks. Only one of ESNS_N, ESNS_0, ESNS_1, and PRG can be asserted. Otherwise, ESNS_E is asserted to indicate this error. 0 Return 0 for all read (read); No meaning (write). 1 Initiate a 0-stressed explicit sense cycle (write).
ESNS_1	Explicit Sense—1 Stressed. Writing 1 to this bit initiates a 1-stressed explicit sense cycle. Reading this bit always returns 0. FSM generates a done signal when the operation completes. This bit is cleared automatically by hardware when sense operation completes. During 1-stressed explicit sense cycles, the EPM_READSENSE1 signal is asserted to the fuse banks. Only one of ESNS_N, ESNS_0, ESNS_1, and PRG can be asserted. Otherwise, ESNS_E is asserted to indicate this error. 0 Return 0 for all read (read); No meaning (write). 1 Initiate a 1-stressed explicit sense cycle (write).
PRG	Program. Writing 1 to this bit initiates a fuse program cycle. Reading this bit always returns 0. FSM generate a done signal when the operation complete. This bit is cleared automatically by hardware when program operation completes. Only one of ESNS_N, ESNS_0, ESNS_1, and PRG can be asserted. Otherwise, ESNS_E is asserted to indicate this error. 0 Return 0 for all read (read); No meaning (write). 1 Initiate a program cycle (write).

18.3.2.6 Upper Address (IIM_UA) Register

The Upper Address (IIM_UA) register contains the top part of the address of the e-Fuse bit to be programmed or the word to be sensed in an explicit sense cycle. Programming is done on a bit-basis, so

the program address is a full-bit address. Sensing is done on a word (8-bit) basis, so the bottom three bits of the address are ignored.

Figure 18-6 shows the bits in the IIM_UA register. Table 18-7 describes the bit fields.

Address: Base + 0x014 Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	A[13:8]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-6. Upper Address Register (IIM_UA) Register

Table 18-7. IIM_UA field descriptions

Field	Description
A[13:8]	The top six bits of the address of the e-Fuse bit to be programmed or the word to be sensed explicitly. The address must be written prior to setting the PRG or ESNS_x bit in IIM_FCTL to initiate the program/sense operation. A[13:11] select the fuse bank. A[10:8] provide the most significant portion of the row address within the bank.

18.3.2.7 Lower Address (IIM_LA) Register

The Lower Address (IIM_LA) register contains the bottom eight bits of the address of the e-Fuse bit to be programmed or word to be explicitly sensed.

Figure 18-7 shows the bits in the IIM_LA register. Table 18-8 describes the bit fields.

Address: Base + 0x018 Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	A[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-7. Lower Address (IIM_LA) Register

Table 18-8. IIM_LA field descriptions

Field	Description
A[7:0]	The bottom eight bits of the address of the e-Fuse bit to be programmed or word to be sensed explicitly. The address must be written prior to setting the PRG or ESNS_x bit in IIM_FCTL to initiate a program or sense operation. A[7:3] provides the least significant portion of the row address. A[2:0] select the bit position within the selected row.

18.3.2.8 Explicit Sense Data Register (IIM_SDAT)

Figure 18-8 shows the bits in the Explicit Sense Data Register (IIM_SDAT). Table 18-9 describes the bit fields.

Address: Base + 0x01C — Access: Supervisor read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	D[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-8. Explicit Sense Data Register (IIM_SDAT)

Table 18-9. IIM_SDAT field descriptions

Field	Description
D[7:0]	The data sensed from the fuses.

On an explicit sense cycle, the data sensed from the fuses is placed in this register at the conclusion of the sense cycle. Software can recognize the conclusion of the sense cycle by the assertion of SNSD in IIM_STAT.

18.3.2.9 Program Protection (IIM_PREG_P) Register

The Program Protection (IIM_PREG_P) register prevents accidental fuse programming. The fuses can be blown only when the value of this register is 0xAA. Software should only program this register to 0xAA while actively blowing fuses. After the program operation is complete, this register should be immediately reprogrammed to a different value.

Figure 18-9 shows the bits in the IIM_PREG_P register. Table 18-10 describes the bit fields.

Address: Base + 0x028 Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PROTECTION_REG[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-9. Program Protection (PRG_P) Register

Table 18-10. PRG_P field descriptions

Field	Description
PROTECTION_REG[7:0]	The fuses can be blown only when the value of this register is 0xAA. Any attempt to program the fuse while the value is other than 0xAA is terminated with error, and the WPE bit is asserted.

18.3.2.10 Divide Factor (IIM_DIVIDE) Register

The Divide Factor (IIM_DIVIDE) register contains the divide factor to generate a 32 kHz clock for programming the fuses from the IPS clock. The unit of IPS clock is MHz.

Figure 18-10 shows the bits in the IIM_DIVIDE register. Table 18-11 describes the bit fields.

Address: Base + 0x03C Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	DIVIDE[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0

Figure 18-10. Divide Factor (IIM_DIVIDE) Register

Table 18-11. IIM_DIVIDE field descriptions

Field	Description
DIVIDE[7:0]	This register provides a divide factor with which IPG_CLK can be divided to an 1 MHz clock. Then the 1 MHz clock is divided to 32 kHz. Its value should be the frequency of IPS clock (unit: MHz), but the minimum is 1. The default value is 0x42.

18.3.2.11 Fuse Bank 1 Protection (IIM_FBAC1) Register

The Fuse Bank 1 Protection (IIM_FBAC1) register corresponds to the first word in fuse bank 1, which holds the fuse bank access protection information. The fuses associated with this register must be sensed out when IIM come out of reset.

Reading these bits returns the fuse state (0 = unblown; 1 = blown) so long as IIM_FBAC1[FBRP] is = (unblown). Disallowed reads always return 0 and cause ERR[RPE] to be set. Writing these bits overrides the values without modifying the fuse elements. Overriding is allowed so long as IIM_FBAC1[FBOP] = 0 (unblown). Disallowed attempts to override are ignored and cause ERR[OPE] to be set. The corresponding fuse elements may be programmed (blown) using the fuse programming sequence, so long as IIM_FBAC1[FBWP] = 0 (unblown). Disallowed attempts to program fuses are ignored and cause ERR[WPE] to be set.

Figure 18-11 shows the bits in the Status register. Table 18-12 describes the bit fields.

Address: Base + 0xC00 Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	FBWP	FBOP	FBRP	FBSP	FB ESP	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	—	—	—	—	—	—	—	—

Figure 18-11. Fuse Bank 1 Protection (IIM_FBAC1) Register

Table 18-12. IIM_FBAC1 field descriptions

Field	Description
FBWP	Fuse Bank Write Protect. Controls whether this fuse bank (Fuse Bank 1) may be programmed. 0 (Unblown) = Fuse bank 1 may be programmed. 1 (Blown) = Fuse bank 1 may not be programmed (it is write-protected).
FBOP	Fuse Bank Override Protect. Controls whether this fuse bank (Fuse Bank 1) may be overridden. 0 (Unblown) = Fuse bank 1 may be overridden. 1 (Blown) = Fuse bank 1 may not be overridden (it is override-protected).
FBRP	Fuse Bank Read Protect. Controls whether this fuse bank (Fuse Bank 1) may be read. 0 (Unblown) = Fuse bank 1 may be read by software. 1 (Blown) = Fuse bank 1 may not be read by software (it is read-protected).
FBSP	
FBESP	Fuse Banks Explicit Sense Protect. Controls whether this fuse bank (Fuse Bank 1) may be explicitly sensed. The state of this fuse controls whether the IIM state machine allow explicit sense cycles (normal, 0-stress, or 1-stress). 0 (Unblown) = Fuse bank 1 may be explicitly sensed by software. 1 (Blown) = Fuse bank 1 may not be explicitly sensed by software (it is sense-protected).

18.3.2.12 Fuse Bank 1 Data (IIM_FB1W1) Register

Figure 18-12 shows the bits in the Fuse Bank 1 Data (IIM_FB1W1) register. Table 18-13 describes the bit fields.

Address: Base + 0xC04 – 0xC7C Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	DATA[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	—	—	—	—	—	—	—	—

Figure 18-12. Fuse Bank 1 Data (IIM_FB1W1) Register

Table 18-13. IIM_FB1W1 field descriptions

Field	Description
DATA[7:0]	Reading these bits returns the fuse state (0 = unblown; 1 = blown) so long as IIM_FBAC1[FBRP] = 0 (unblown). Disallowed reads always return 0 and cause ERR[RPE] to be set. Writing these bits overrides the values without modifying the fuse elements. Overriding is allowed so long as IIM_FBAC1[FBOP] = 0. Disallowed attempts to override are ignored and cause ERR[OPE] to be set. The corresponding fuse elements may be programmed (blown) using the fuse programming sequence, so long as IIM_FBAC1[FBWP] = 0. Disallowed attempts to program fuses are ignored and cause ERR[WPE] to be set.

18.4 Functional Description

The IIM consists of a master controller, a fuse value shadow cache, and a set of registers to hold the values of signals visible outside the module. Two 256-bit fuse banks are implemented on the MPC5125.

Program operations are done on a bit basis. For programming, the external FUSE programming supply A_{VDD_FUSEWR} must be applied.

18.4.1 Fuse Bank 1

Fuse bank 1 (256 bits) is available for user data.

Chapter 19
Inter-Integrated Circuit (I^2C)

19.1 Overview

The Inter-Integrated Circuit (I^2C) interface is a two-wire, bidirectional serial bus that provides a simple, efficient method for data exchange among devices. The MPC5125 contains three identical and independent I2C modules.

The I^2C module operates up to a maximum bus load and timing of 400 kbit/s. Also, I^2C modules are capable of operating at higher baud rates, up to a maximum frequency equal to IPS_clock/20 with reduced bus loading (where, the value 20 is the minimum SCL Divider in the block).

A maximum bus capacitance of 400 pF limits the maximum communication length and number of possibly connected devices. The module can operate as fast as 400 kbit/s. This bus is suitable for applications requiring occasional communications over a short distance among a number of devices. It also provides flexibility, allowing more devices to be connected to the bus for further expansion and system development.

I^2C is a true multi-master bus including collision detection and arbitration to prevent data corruption if two or more masters attempt to control the bus simultaneously. This feature provides the capability for complex applications with multi-processor control. It may also be used for rapid testing and alignment of end products via external connections to an assembly-line computer. Figure 19-1 shows a block diagram of the I^2C module.

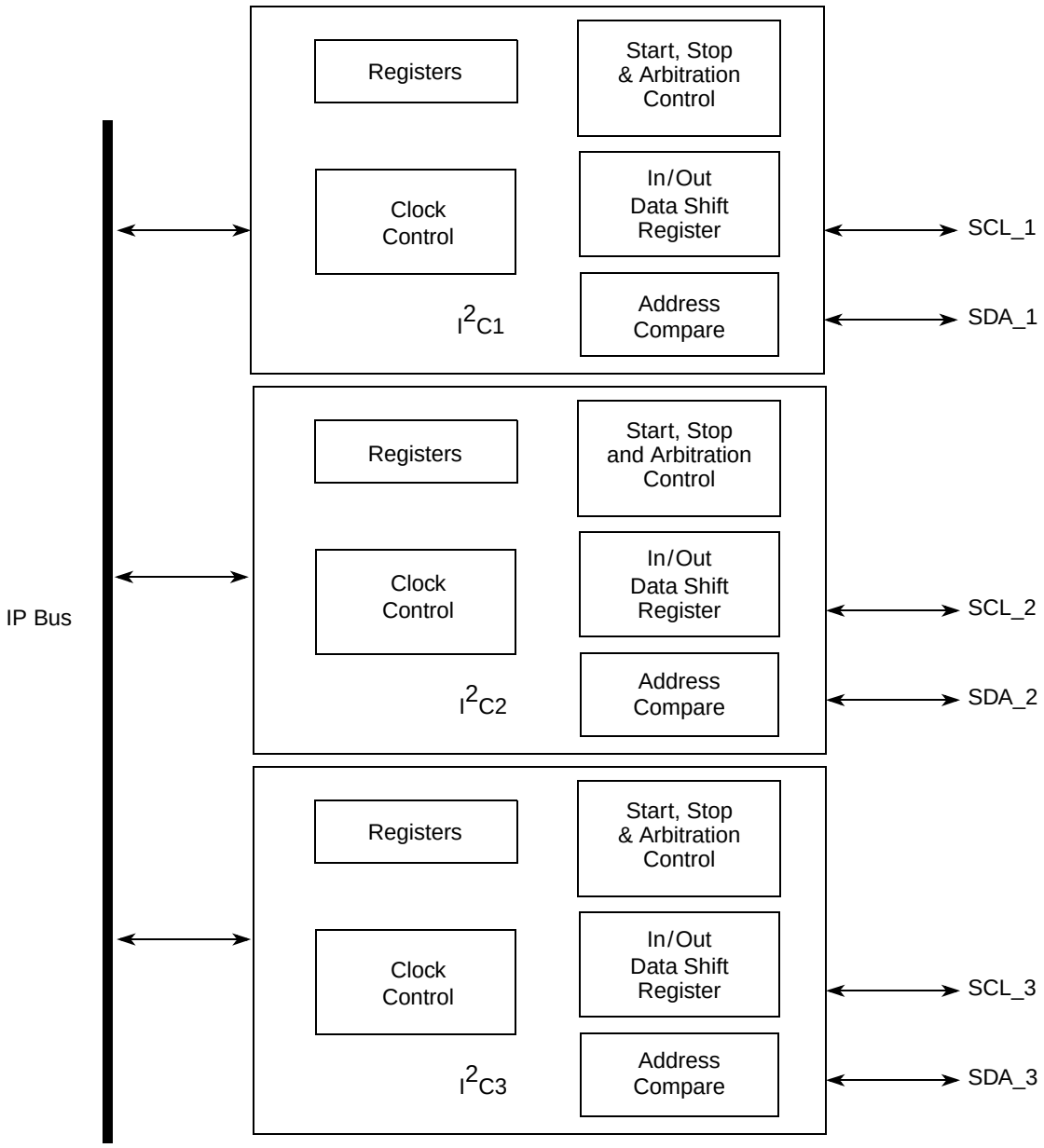


Figure 19-1. Block Diagram – I^{2}C Module

19.1.1 Features

The I^{2}C module has these key features:

- Compatible with I^{2}C bus standard version 2.1
- Multi-master operation
- Software programmable for different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer

- Arbitration loss with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- Programmable glitch filter

19.1.2 I²C Controller

The I²C is a simple bidirectional two-wire bus for efficient device-to-device communication. The two wires, serial data line (SDA) and serial clock line (SCL), carry information between this module and other devices connected to the bus. A unique address recognizes each device. Also, each device can operate as transmitter or receiver, depending on the function of the device. In addition to the transmitters and receivers, devices can be masters or slaves. A master is the device that initiates a data transfer on the bus and generates clock signals to permit that transfer. At that time, any device addressed is considered a slave. See Table 19-1.

Table 19-1. I²C Terminology

Term	Description
Transmitter	Device that sends data to bus.
Receiver	Device that receives data from bus.
Master	Device that initiates transfer, generates SCL, and terminates transfer.
Slave	Device that is addressed by master.

Standard communication usually has four functional areas:

- START signal
- Slave address transmission
- Data transfer
- STOP signal

These activities are briefly described in the following sections.

19.1.3 START Signal

A START signal is a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer and wakes up all slaves. Each data transfer may contain several data bytes.

When the bus is free (no master device is engaging the bus), SCL and SDA lines are at a logical high. A master sends a START signal to initiate communications.

19.1.4 STOP Signal

A STOP signal is a low-to-high transition of SDA while SCL is high.

The master generates a STOP signal to terminate communication, which frees the bus. The master can generate a STOP even if the slave has generated an acknowledge, at which point the slave must release the bus.

The master can generate a repeated start and address for other devices. At this time, the bus remains busy if a repeated start is generated instead of a stop.

19.1.4.1 Slave Address Transmission

The first byte of data transferred by the master immediately after a START signal is the slave address. This is a 7-bit calling address followed by a R/$\overline{W}$ bit. The R/$\overline{W}$ bit tells the slave the desired direction of data transfer.

- 0 = Master writes data (W), becomes transmitter
- 1 = Master reads data (R), becomes receiver

Only a slave with a calling address matching the address transmitted by the master responds by sending back an acknowledge bit. This is done by pulling SDA low at the ninth clock as shown in Figure 19-2.

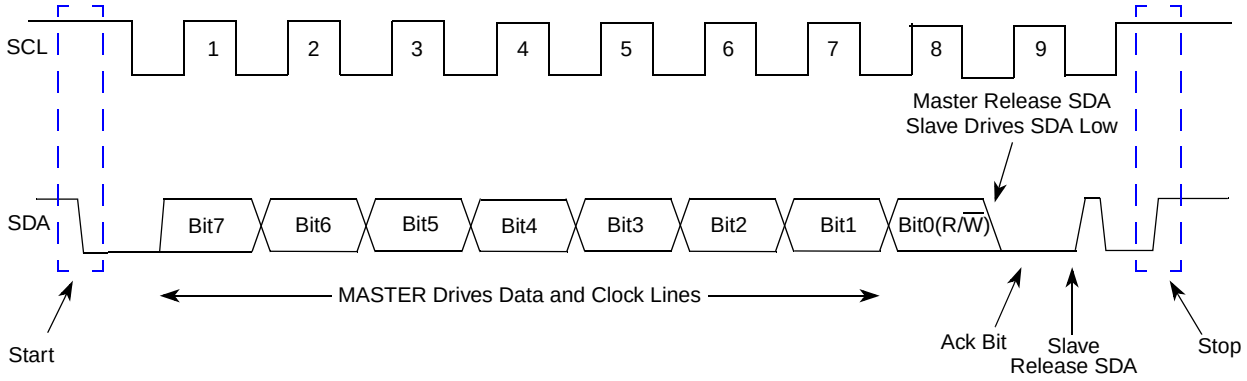


Figure 19-2. Timing Diagram–Start, Address Transfer and Stop Signal

19.1.4.2 Data Transfer

Data transfer proceeds byte-by-byte in a direction specified by the R/$\overline{W}$ bit sent by the calling master. Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high.

There is one clock pulse on SCL for each data bit. The MSB is transferred first. Each data byte must be followed by an acknowledge bit signalled from the receiving device by pulling SDA low at the ninth clock. One complete 8-bit data byte transfer needs nine clock pulses as shown in Figure 19-3.

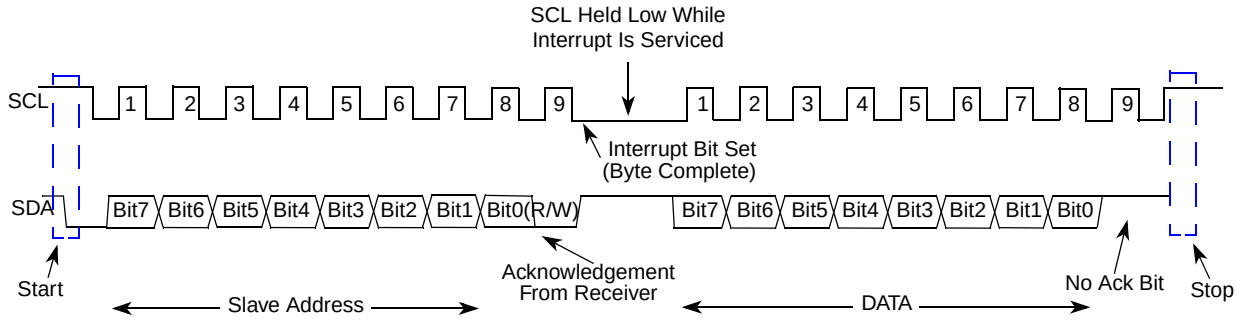


Figure 19-3. Timing Diagram–Start, Address Transfer and Stop Signal

19.1.5 Acknowledge

Figure 19-4 shows the transmitter releasing the SDA line after bit 0 is transmitted. All devices driving the SDA line must have an open-collector configuration. A pull-up register is required to ensure the SDA line goes to a logic 1 when not driven by the master device. The receiver pulls the SDA line low during the acknowledge clock pulse to signal correct reception of the data.

If a slave-receiver does not acknowledge the byte transfer, SDA must be left HIGH by the slave. The master then generates a STOP condition to abort the transfer.

If a master-receiver does not acknowledge the slave transmitter after a byte transmission, it means End-Of-Data (EOD) to the slave. The slave then releases the SDA line for the master to generate a STOP or START signal.

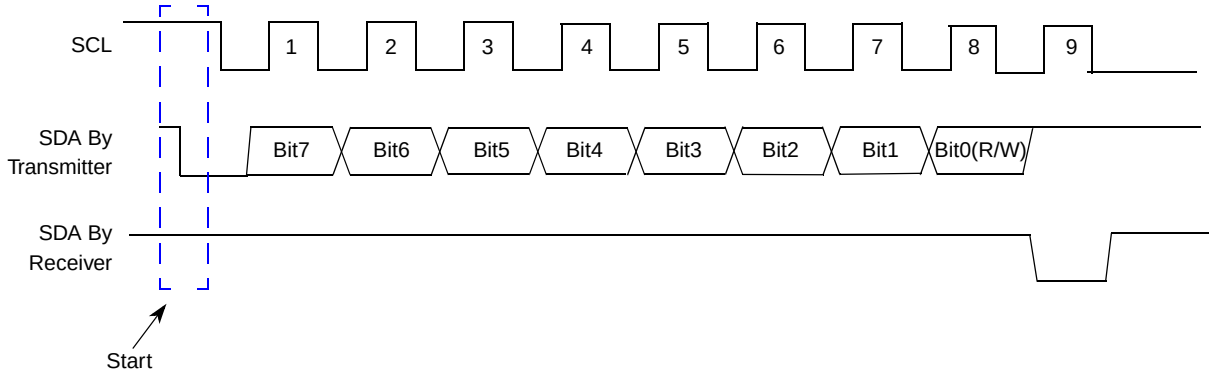


Figure 19-4. Timing Diagram – Receiver Acknowledgement

19.1.5.1 Repeated Start

A repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. The master uses this to communicate with another slave or with the same slave in a different mode without releasing the bus.

Various combinations of read/write formats are possible. Figure 19-5 shows examples of:

- The master-transmitter transmitting to a slave-receiver. The transfer direction is not changed.

- The master reading a slave immediately after first byte. At the moment of the first acknowledge, the master-transmitter becomes a master-receiver and the slave-receiver becomes a slave-transmitter.
- The START condition and slave address both repeated using the repeated START signal. This communicates with the same slave in a different mode without releasing the bus. The master transmits data to the slave first, and then the master reads data from the slave by reversing the R/$\overline{W}$ bit.

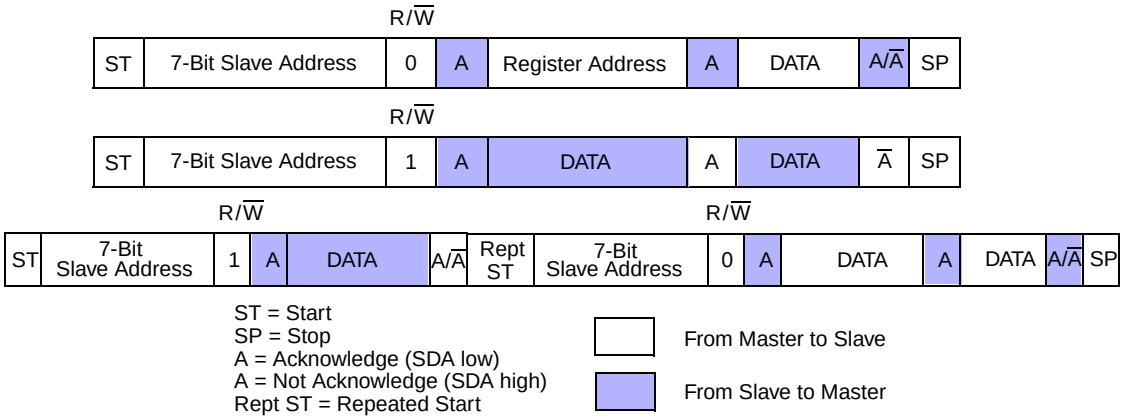


Figure 19-5. Data Transfer, Combined Format

19.1.5.2 Clock Synchronization

I^2C is a true multi-master bus. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock.

Because wire-AND logic is used on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices start counting their low period. After a device clock goes low, it holds the SCL line low until the clock high state is reached. However, the change of low-to-high in this device clock may not change the SCL line state if another device clock remains within its low period. Therefore, the synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time. See Figure 19-6.

When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. No difference exists between device clocks and the SCL line state. All devices start counting their high periods. The first device to complete its high period pulls the SCL line low again.

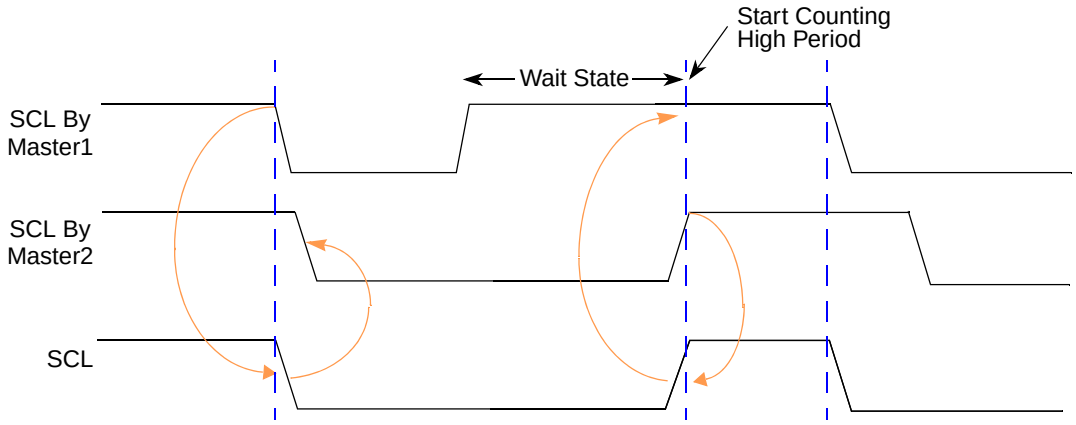


Figure 19-6. Timing Diagram–Clock Synchronization

19.1.6 Arbitration

A data arbitration procedure determines the relative priority of contending masters. A bus master loses arbitration if it transmits logic 1 while another master transmits logic 0. Losing masters immediately switch to slave-receive mode and stop driving SDA output. In this case, transition from master to slave mode does not generate a STOP condition. A status bit is hardware set to indicate loss of arbitration. Figure 19-7 shows an example of two masters arbitrating for the I^2C bus.

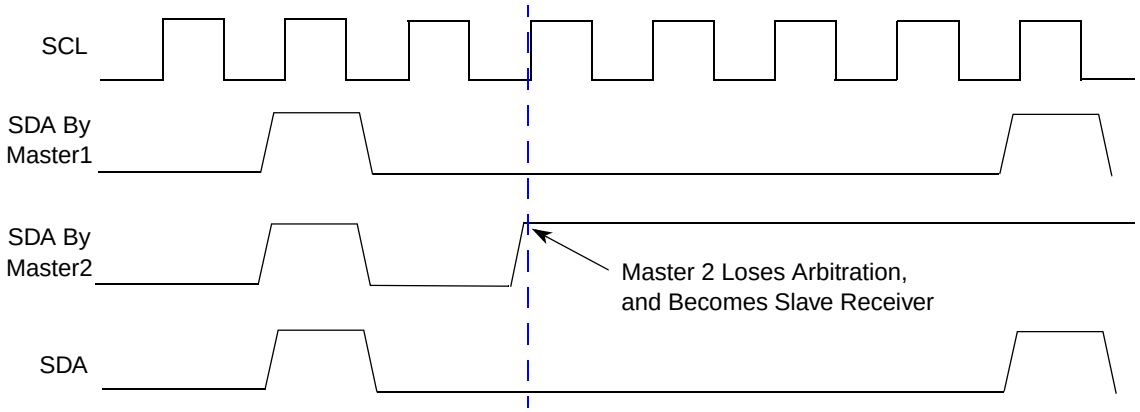


Figure 19-7. Timing Diagram – Arbitration Procedure

19.2 External Signal Description

Table 19-2. Signal Properties

Name	Function	I/O	Reset	Pull Up
SCL	I^2C serial clock line, bidirectional	I/O	1	Yes
SDA	I^2C serial data line, bidirectional	I/O	1	Yes

19.3 Memory Map and Register Definition

The I^2C is controlled by 17 32-bit registers. The registers are located at an offset from I^2C base address, I^2C_BASE, which is memory mapped on the IP Bus address. The I^2C1 base address (I^2C1_base_address) is I^2C_BASE + 0x00. The I^2C2 base address (I^2C2_base_address) is I^2C_BASE + 0x20. The I^2C3 base address (I^2C3_base_address) is I^2C_BASE + 0x40. Each of their own private registers addresses are relative to their base offset. There are two common registers for the three I^2C modules: I^2C interrupt control register and glitch filter control register.

The I^2C Interface registers are provided below:

Table 19-3. I^2C block memory map

Offset from I2C_BASE[1] (0xFF40_1700)	Register	Access[2]	Reset Value[3]	Section/Page
0x00	I2C_MADR1 – I^2C1 Address Register	R/W	0x0000_0000	19.3.1.1/19-477
0x04	I2C_MFDR1 – I^2C1 Frequency Divider Register	R/W	0x0000_0000	19.3.1.2/19-478
0x08	I2C_MCR1 – I^2C1 Control Register	R/W	0x0000_0000	19.3.1.3/19-485
0x0C	I2C_MSR1 – I^2C1 Status Register	R/W	0x8000_0000	19.3.1.4/19-486
0x10	I2C_MDR1 – I^2C1 Date I/O Register	R/W	0x0000_0000	19.3.1.5/19-488
0x20	I2C_MADR2 – I^2C2 Address Register	R/W	0x0000_0000	19.3.1.1/19-477
0x24	I2C_MFDR2 – I^2C2 Frequency Divider Register	R/W	0x0000_0000	19.3.1.2/19-478
0x28	I2C_MCR2 – I^2C2 Control Register	R/W	0x0000_0000	19.3.1.3/19-485
0x2C	I2C_MSR2 – I^2C2 Status Register	R/W	0x8000_0000	19.3.1.4/19-486
0x30	I2C_MDR2 – I^2C2 Date I/O Register	R/W	0x0000_0000	19.3.1.5/19-488
0x40	I2C_MADR3 – I^2C3 Address Register	R/W	0x0000_0000	19.3.1.1/19-477
0x44	I2C_MFDR3 – I^2C3 Frequency Divider Register	R/W	0x0000_0000	19.3.1.2/19-478
0x48	I2C_MCR3 – I^2C3 Control Register	R/W	0x0000_0000	19.3.1.3/19-485
0x4C	I2C_MSR3 – I^2C3 Status Register	R/W	0x8000_0000	19.3.1.4/19-486
0x50	I2C_MDR3 – I^2C3 Date I/O Register	R/W	0x0000_0000	19.3.1.5/19-488

Table 19-3. I²C block memory map (Continued)

Offset from I2C_BASE[1] (0xFF40_1700)	Register	Access[2]	Reset Value[3]	Section/Page
0x54–0x5F	Reserved			
0x60	I2C_ICR – I²C Interrupt Control Register	R/W	0x1500_0000	19.3.1.6/19-489
0x64	I2C_MIFR – I²C Filter Register	R/W	0x0000_0000	19.3.1.7/19-490
0x68–0xFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, "System Configuration and Memory Map (XLBMEN + Mem Map)."

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

3 In this column, the symbol "U" indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

19.3.1 Register Descriptions

19.3.1.1 I²C Address Register (I2C_MADR*n*)

Address: Base + 0x00 (I2C_MADR)
Base + 0x20 (I2C_MADR2)
Base + 0x40 (I2C_MADR3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADR[7:1]							0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-8. I²C Address Register (I2C_MADR*n*)

Table 19-4. I2C_MADR*n* field descriptions

Field	Description
ADR[7:1]	Bits 0 to 6 contains the address I²C responds to when addressed as a slave. This is not the address sent on the bus during address transfer.

19.3.1.2 I²C Frequency Divider Register (I2C_MFDR*n*)

Address: Base + 0x04 (I2C_MFDR1)
Base + 0x24 (I2C_MFDR2)
Base + 0x44 (I2C_MFDR3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FDR[7:0]								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-9. I²C Address Register (I2C_MFDR*n*)

Table 19-5. I2C_MFDR*n* field descriptions

Field	Description
FDR[7:6]	These two bits act as a prescale divider of the input module clock.
FDR[5:0]	This field preschools the clock for bit-rate selection.

The frequency divide register determines the SCL or serial bit-clock frequency. Table 19-6 must be used to select FDR bits that produce an appropriate SCL. The following relationships illustrate the connection between Table 19-6 and the signals in the I²C timing specification.

1. SCL (in kHz) = (1/1000) × [IPS clock speed (in Hz)]/(SCL Period)
2. SDA Hold Time (in µs) = 1000 × (SDA Hold/SCL Period)/[SCL (in kHz)]
3. SCL Hold Time of START (in µs) = 1000 × (SDA Hold of START/SCL Period)/[SCL (in kHz)]
4. SCL Hold Time of STOP (in µs) = 1000 × (SDA Hold of STOP/SCL Period))/[SCL (in kHz)]

Figure 19-10 illustrates the relationship between the IPS clock and the I²C signals.

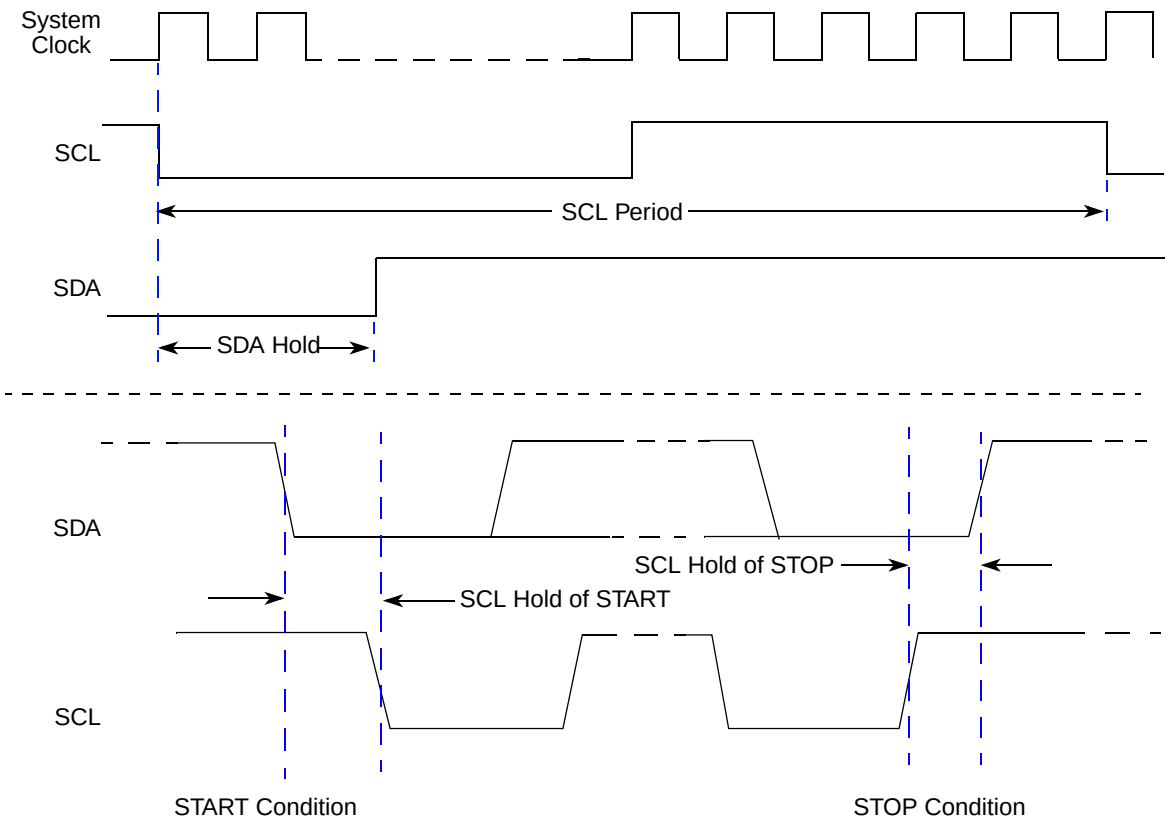


Figure 19-10. Timing Diagram of I²C Signal Relationships

For standard mode I²C, the I²C specification states:

$$\text{(SCL} \leq \text{100 kHz)} \qquad \textit{Eqn. 19-1}$$

and

$$\text{(0.3 µs} \leq \text{SDA Hold Time} \leq \text{3.45 µs)} \qquad \textit{Eqn. 19-2}$$

and

$$\text{(SCL Hold of START} \geq \text{4 µs)} \qquad \textit{Eqn. 19-3}$$

and

$$\text{(SCL Hold of STOP} \geq \text{4 µs)} \qquad \textit{Eqn. 19-4}$$

This means the system programmer must choose SCL Period, SDA Hold, SCL Hold of START, and SCL Hold of STOP from Table 19-6 to satisfy Equation 19-5 through Equation 19-8.

$$\text{SCL Period)} \geq \text{(1/100,000)} \times \text{[IPS clock speed (in Hz)]} \qquad \textit{Eqn. 19-5}$$

and

$$\text{(0.0003)} \times \text{[SCL (in kHz)]} \times \text{(SCL Period)} \leq \text{SDA Hold} \leq \text{(0.00345)} \times \text{[SCL (in kHz)]} \times \text{(SCL Period)} \qquad \textit{Eqn. 19-6}$$

and

$$\text{SCL Hold of START} \geq (0.004) \times [\text{SCL (in kHz)}] \times (\text{SCL Period}) \quad \textit{Eqn. 19-7}$$

and

$$\text{SCL Hold of STOP} \geq (0.004) \times [\text{SCL (in kHz)}] \times (\text{SCL Period}) \quad \textit{Eqn. 19-8}$$

In this case, the simplest strategy for the system programmer to use is:

1. Identify all rows of Table 19-6 where SCL period satisfies criteria in Equation 19-5. This set of rows limits the choices of SCL allowed for this particular IPS clock.
2. Calculate the SCL associated with these rows according to Equation 19-1 and decide which speeds are acceptable (fast enough or slow enough) for the system.
3. Find the subset of those rows associated with the acceptable I^2C clock speeds such that SDA hold satisfies criteria in Equation 19-6.
4. Choose the preferred FDR setting from among the subset that meets Equation 19-5 and Equation 19-6.
5. Check that the preferred FDR setting also satisfies Equation 19-7 and Equation 19-8. Usually, it does. If not, choose a different FDR setting that meets Equation 19-5, Equation 19-6, Equation 19-7, and Equation 19-8.

Likewise, for fast mode I^2C, it must also meet the fast-mode I^2C bus specification.

$$(\text{SCL} \leq 400 \text{ kHz}) \quad \textit{Eqn. 19-9}$$

and

$$(0.3 \text{ us} \leq \text{SDA Hold Time} \leq 0.9 \text{ µs}) \quad \textit{Eqn. 19-10}$$

and

$$(\text{SCL Hold of START} \geq 0.6 \text{ µs}) \quad \textit{Eqn. 19-11}$$

and

$$(\text{SCL Hold of STOP} \geq 0.6 \text{ µs}) \quad \textit{Eqn. 19-12}$$

That means the system programmer must choose SCL Period, SDA hold, SCL hold of START, and SCL hold of STOP from Table 19-6 to satisfy Equation 19-9 through Equation 19-12:

$$(\text{SCL Period}) \geq (1/400{,}000) \times [\text{IPS clock speed (in Hz)}] \quad \textit{Eqn. 19-13}$$

and

$$(0.0003) \times [\text{SCL (in kHz)}] \times (\text{SCL Period}) \leq \text{SDA Hold} \leq (0.0009) \times [\text{SCL (in kHz)}] \times (\text{SCL Period}) \quad \textit{Eqn. 19-14}$$

and

$$\text{SCL Hold of START} \geq (0.0006) \times [\text{SCL (in kHz)}] \times (\text{SCL Period}) \quad \textit{Eqn. 19-15}$$

and

$$\text{SCL Hold of STOP} \geq (0.0006) \times [\text{SCL (in kHz)}] \times (\text{SCL Period}) \quad \textit{Eqn. 19-16}$$

In this case, the strategy to choose FDR value is like the one for standard mode device, but based on the different timing requirements.

In Table 19-6, the SCL Divider has the same meaning as SCL Period shown in Figure 19-10, which uses the IPS clock as a time unit. For example, from the table, if the SCL Divider equals 20, the SCL Period is equal to 20 IPS clocks.

Table 19-6. I^{2}C Frequency Divider Bit Selection

FDR Value	SCL Divider	SDA Hold	SCL Hold of START	SCL Hold of STOP	FDR Value	SCL Divider	SDA Hold	SCL Hold of START	SCL Hold of STOP
0x20	20	7	6	11	0x13	480	65	238	241
0x21	22	7	7	12	0x4F	480	66	236	242
0x22	24	8	8	13	0x37	512	65	254	257
0x23	26	8	9	14	0x73	512	66	252	258
0x24	28	7	10	15	0xAF	512	68	248	260
0x00	28	9	10	15	0xEF	512	68	248	260
0x01	30	9	11	16	0x8B	512	84	232	260
0x25	32	7	12	17	0xCB	512	84	232	260
0x02	34	10	13	18	0x14	576	97	286	289
0x26	36	9	14	19	0x50	576	98	284	290
0x27	40	9	16	21	0x8C	576	100	280	292
0x03	40	10	16	21	0xCC	576	100	280	292
0x60	40	14	12	22	0x38	640	65	318	321
0x04	44	11	18	23	0x74	640	66	316	322
0x61	44	14	14	24	0xB0	640	68	312	324
0x28	48	9	18	25	0xF0	640	68	312	324
0x05	48	11	20	25	0x15	640	97	318	321
0x62	48	16	16	26	0x51	640	98	316	322
0x63	52	16	18	28	0x8D	640	100	312	324
0x29	56	9	22	29	0xCD	640	100	312	324
0x06	56	13	24	29	0x39	768	65	382	385
0x64	56	14	20	30	0x75	768	66	380	386
0x40	56	18	20	30	0xB1	768	68	376	388
0x41	60	18	22	32	0xF1	768	68	376	388
0x2A	64	13	26	33	0x16	768	129	382	385
0x65	64	14	24	34	0x52	768	130	380	386
0x07	68	13	30	35	0x8E	768	132	376	388

Table 19-6. I²C Frequency Divider Bit Selection (Continued)

FDR Value	SCL Divider	SDA Hold	SCL Hold of START	SCL Hold of STOP	FDR Value	SCL Divider	SDA Hold	SCL Hold of START	SCL Hold of STOP
0x42	68	20	26	36	0xCE	768	132	376	388
0x2B	72	13	30	37	0x3A	896	129	446	449
0x66	72	18	28	38	0x76	896	130	444	450
0x2C	80	9	38	41	0xB2	896	132	440	452
0x08	80	17	34	41	0xF2	896	132	440	452
0x67	80	18	32	42	0x17	960	129	478	481
0x43	80	20	32	42	0x53	960	130	476	482
0xA0	80	28	24	44	0x8F	960	132	472	484
0xE0	80	28	24	44	0xCF	960	132	472	484
0x09	88	17	38	45	0x3B	1024	129	510	513
0x44	88	22	36	46	0x77	1024	130	508	514
0xA1	88	28	28	48	0xB3	1024	132	504	516
0xE1	88	28	28	48	0xF3	1024	132	504	516
0x2D	96	9	46	49	0x18	1152	193	574	577
0x68	96	18	36	50	0x54	1152	194	572	578
0x45	96	22	40	50	0x90	1152	196	568	580
0xA2	96	32	32	52	0xD0	1152	196	568	580
0xE2	96	32	32	52	0x3C	1280	129	638	641
0x0A	104	21	46	53	0x78	1280	130	636	642
0xA3	104	32	36	56	0xB4	1280	132	632	644
0xE3	104	32	36	56	0xF4	1280	132	632	644
0x2E	112	17	54	57	0x19	1280	193	638	641
0x69	112	18	44	58	0x55	1280	194	636	642
0x46	112	26	48	58	0x91	1280	196	632	644
0xA4	112	28	40	60	0xD1	1280	196	632	644
0xE4	112	28	40	60	0x3D	1536	129	766	769
0x80	112	36	40	60	0x79	1536	130	764	770
0xC0	112	36	40	60	0xB5	1536	132	760	772
0x81	120	36	44	64	0xF5	1536	132	760	772
0xC1	120	36	44	64	0x1A	1536	257	766	769
0x2F	128	17	62	65	0x56	1536	258	764	770
0x0B	128	21	58	65	0x92	1536	260	760	772

Table 19-6. I^2C Frequency Divider Bit Selection (Continued)

FDR Value	SCL Divider	SDA Hold	SCL Hold of START	SCL Hold of STOP	FDR Value	SCL Divider	SDA Hold	SCL Hold of START	SCL Hold of STOP
0x6A	128	26	52	66	0xD2	1536	260	760	772
0xA5	128	28	48	68	0x3E	1792	257	894	897
0xE5	128	28	48	68	0x7A	1792	258	892	898
0x47	136	26	60	70	0xB6	1792	260	888	900
0x82	136	40	52	72	0xF6	1792	260	888	900
0xC2	136	40	52	72	0x1B	1920	257	958	961
0x0C	144	25	70	73	0x57	1920	258	956	962
0x6B	144	26	60	74	0x93	1920	260	952	964
0xA6	144	36	56	76	0xD3	1920	260	952	964
0xE6	144	36	56	76	0x3F	2048	257	1022	1025
0x30	160	17	78	81	0x7B	2048	258	1020	1026
0x6C	160	18	76	82	0xB7	2048	260	1016	1028
0x0D	160	25	78	81	0xF7	2048	260	1016	1028
0x48	160	34	68	82	0x1C	2304	385	1150	1153
0xA7	160	36	64	84	0x58	2304	386	1148	1154
0xE7	160	36	64	84	0x94	2304	388	1144	1156
0x83	160	40	64	84	0xD4	2304	388	1144	1156
0xC3	160	40	64	84	0x7C	2560	258	1276	1282
0x49	176	34	76	90	0xB8	2560	260	1272	1284
0x84	176	44	72	92	0xF8	2560	260	1272	1284
0xC4	176	44	72	92	0x1D	2560	385	1278	1281
0x31	192	17	94	97	0x59	2560	386	1276	1282
0x6D	192	18	92	98	0x95	2560	388	1272	1284
0x0E	192	33	94	97	0xD5	2560	388	1272	1284
0xA8	192	36	72	100	0x7D	3072	258	1532	1538
0xE8	192	36	72	100	0xB9	3072	260	1528	1540
0x85	192	44	80	100	0xF9	3072	260	1528	1540
0xC5	192	44	80	100	0x1E	3072	513	1534	1537
0x4A	208	42	92	106	0x5A	3072	514	1532	1538
0x32	224	33	110	113	0x96	3072	516	1528	1540
0x6E	224	34	108	114	0xD6	3072	516	1528	1540
0xA9	224	36	88	116	0x7E	3584	514	1788	1794

Table 19-6. I²C Frequency Divider Bit Selection (Continued)

FDR Value	SCL Divider	SDA Hold	SCL Hold of START	SCL Hold of STOP	FDR Value	SCL Divider	SDA Hold	SCL Hold of START	SCL Hold of STOP
0xE9	224	36	88	116	0xBA	3584	516	1784	1796
0x86	224	52	96	116	0xFA	3584	516	1784	1796
0xC6	224	52	96	116	0x1F	3840	513	1918	1921
0x0F	240	33	118	121	0x5B	3840	514	1916	1922
0x33	256	33	126	129	0x97	3840	516	1912	1924
0x6F	256	34	124	130	0xD7	3840	516	1912	1924
0x4B	256	42	116	130	0x7F	4096	514	2044	2050
0xAA	256	52	104	132	0xBB	4096	516	2040	2052
0xEA	256	52	104	132	0xFB	4096	516	2040	2052
0x87	272	52	120	140	0x5C	4608	770	2300	2306
0xC7	272	52	120	140	0x98	4608	772	2296	2308
0x10	288	49	142	145	0xD8	4608	772	2296	2308
0x4C	288	50	140	146	0xBC	5120	516	2552	2564
0xAB	288	52	120	148	0xFC	5120	516	2552	2564
0xEB	288	52	120	148	0x5D	5120	770	2556	2562
0x34	320	33	158	161	0x99	5120	772	2552	2564
0x70	320	34	156	162	0xD9	5120	772	2552	2564
0xAC	320	36	152	164	0xBD	6144	516	3064	3076
0xEC	320	36	152	164	0xFD	6144	516	3064	3076
0x11	320	49	158	161	0x5E	6144	1026	3068	3074
0x4D	320	50	156	162	0x9A	6144	1028	3064	3076
0x88	320	68	136	164	0xDA	6144	1028	3064	3076
0xC8	320	68	136	164	0xBE	7168	1028	3576	3588
0x89	352	68	152	180	0xFE	7168	1028	3576	3588
0xC9	352	68	152	180	0x5F	7680	1026	3836	3842
0x35	384	33	190	193	0x9B	7680	1028	3832	3844
0x71	384	34	188	194	0xDB	7680	1028	3832	3844
0xAD	384	36	184	196	0xBF	8192	1028	4088	4100
0xED	384	36	184	196	0xFF	8192	1028	4088	4100
0x12	384	65	190	193	0x9C	9216	1540	4600	4612
0x4E	384	66	188	194	0xDC	9216	1540	4600	4612
0x8A	416	84	184	212	0x9D	10240	1540	5112	5124

Table 19-6. I²C Frequency Divider Bit Selection (Continued)

FDR Value	SCL Divider	SDA Hold	SCL Hold of START	SCL Hold of STOP	FDR Value	SCL Divider	SDA Hold	SCL Hold of START	SCL Hold of STOP
0xCA	416	84	184	212	0xDD	10240	1540	5112	5124
0x36	448	65	222	225	0x9E	12288	2052	6136	6148
0x72	448	66	220	226	0xDE	12288	2052	6136	6148
0xAE	448	68	216	228	0x9F	15360	2052	7672	7684
0xEE	448	68	216	228	0xDF	15360	2052	7672	7684

19.3.1.3 I²C Control Register (I2C_MCR*n*)

Address: Base + 0x08 (I2C_MCR1)
Base + 0x28 (I2C_MCR2)
Base + 0x48 (I2C_MCR3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EN	IEN	STA	TX	TXAK	RSTA	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-11. I²C Control Register (I2C_MCR*n*)

Table 19-7. I2C_MCR*n* field descriptions

Field	Description
EN	I²C Enable. Bit controls software reset of entire I²C module. If I²C module is enabled in the middle of a byte transfer, interface behaves as follows: Slave mode ignores current bus transfer and starts operating when a subsequent start condition is detected. Master mode is not aware if bus is busy. If a start cycle is initiated, current bus cycle may become corrupt. Ultimately, this results in the current bus master or I²C module losing arbitration, after which bus operation returns to normal. 0 Module is reset and disabled. This is the Power-ON reset. When low the interface is held in reset, but registers can continue to be accessed. 1 I²C module is enabled. Bit must be set before other CR bits have any effect.
IEN	I²C Interrupt Enable 0 Interrupts from I²C module are disabled. This does not clear currently pending interrupt conditions. 1 Interrupts from I²C module are enabled. An I²C interrupt occurs, provided the status register IF bit is also set.
STA	Master/Slave Mode Select. Bit clears on reset. When bit changes from 0 to 1, a START signal is generated on the bus and master mode is selected. When bit changes from 1 to 0, a STOP signal is generated and operation mode changes from master to slave. STA is cleared without generating a STOP signal when the master loses arbitration. 0 Slave Mode 1 Master Mode

Table 19-7. I2C_MCR*n* field descriptions (Continued)

Field	Description
TX	Transmit/Receive Mode Select. Bit selects master/slave transfer direction. When addressed as slave, software should set according to status register SRW bit. When in master mode, bit should be set according to type of transfer required. For address cycles, bit is always high. 0 Receive 1 Transmit
TXAK	Transmit Acknowledge Enable. Bit specifies value driven to SDA during acknowledge cycles for master and slave receivers. Values are used only when I^2C is a receiver, not a transmitter. 0 Acknowledge signal is sent to bus at 9th clock bit after receiving 1 Byte of data. 1 No acknowledge signal response is sent (i.e., acknowledge bit = 1)
RSTA	Repeat Start. Writing 1 to this bit generates a repeated START condition on the bus, provided it is the current bus master. Bit is always read low. If the bus is owned by another master, attempting a repeated start at the wrong time results in loss of arbitration. 1 Generate repeat start cycle

19.3.1.4 I^2C Status Register (I2C_MSR*n*)

The I^2C status register (I2C_MSR*n*) is read-only with the exception of the AL, AKF, and IF bits, which are software clearable by writing a 0. Writing a 1 to the AL, AKF, and IF bits I2C does nothing. To avoid clearing these three flags unintentionally, force a cleared bit to 0 and set a non-cleared bit to 1. For example, to clear the AL bit, only the AL bit must be set to 0 (AL = 0). The other two flags should be equal to 1 (AKF = 1, IF = 1).

Address: Base + 0x0C (I2C_MSR1)
Base + 0x2C (I2C_MSR2)
Base + 0x4C (I2C_MSR3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CF	AAS	BB	AL	AKF	SRW	IF	RXAK	0	0	0	0	0	0	0	0
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-12. I^2C Status Register (I2C_MSR*n*)

Table 19-8. I2C_MSR*n* field descriptions

Field	Description
CF	Data transferring. Bit clears while 1 byte of data is being transferred. This bit is set by the falling edge of ninth clock of a byte transfer. 0 Transfer in progress. 1 Transfer complete.

Table 19-8. I2C_MSR*n* field descriptions (Continued)

Field	Description
AAS	Addressed As Slave. Bit sets when its own specific address (I^2C Address Register) is matched with the calling address. The CPU is interrupted provided IEN is set. The CPU needs to check the SRW bit and set its Tx/Rx mode accordingly. Writing to the I^2C control register clears this bit. 0 Not addressed. 1 Addressed as a slave.
BB	Bus Busy. Bit indicates bus status. When a START signal is detected, BB is set. If a STOP signal is detected, it is cleared. 0 Bus is idle. 1 Bus is busy.
AL	Arbitration Lost.Hardware sets the bit when the arbitration procedure is lost. Arbitration is lost in the following circumstances: • SDA sampled low when master drives high during an address or data Tx cycle. • SDA sampled low when master drives high during a data Rx cycle acknowledge bit. • Start cycle is attempted when bus is busy. • A repeated start cycle is requested in slave mode. • Stop condition is detected when not requested by master. 0 No arbitration lost. 1 Arbitration lost. Software must clear this bit by writing 0 in the interrupt routine after detecting arbitration lost and interrupt flag (IF) bit is asserted.
AKF	Acknowledge Cycle Falling Edge when Arbitration Lost and Addressed as Slave. Hardware sets the bit upon the falling edge of the acknowledge cycle after arbitration has been lost and addressed as slave. In this specific case, the interrupt (IF = 1) is really the second one set by the hardware. Section 19.5.3, “Special Note on AKF.” The software must use this bit to distinguish if the interrupt is the first one (set upon rising edge of acknowledge cycle) or the second one (set upon falling edge of acknowledge cycle). The software should only take action for AL and AAS if the interrupt is the second one, the traditional time for the interrupt. 0 First interrupt on rising edge of acknowledge cycle – software should not take AL and AAS action (see later section for typical software flow diagram). 1 Second interrupt on falling edge of acknowledge – software should take AL and AAS action. If AKF bit is set to 1 by hardware, software must clear this bit by writing it 0 in the interrupt routine.
SRW	Slave Read/Write. When set, bit indicates the R/W command bit value of the calling address sent from the master. **Note:** Bit is valid only when I^2C is in slave mode, a complete address transfer occurred with an address match, and no other transfers were initiated. Checking this bit, the CPU can select slave Tx/Rx mode according to the master command. 0 Slave receive, master writing to slave. 1 Slave transmit, master reading from slave.
IF	I^2C Interrupt. Sets when an interrupt is pending. If IEN is set, a processor interrupt request is generated. IF sets when one of the following events occurs: • Complete 1-Byte transfer (set at falling edge of ninth clock). • A Rx calling address matches its own specific address in slave mode. • Arbitration is lost. 0 No interrupt is generated. 1 An interrupt is generated under the above listed condition. Software must clear this interrupt bit by writing it 0 in the interrupt routine.

Table 19-8. I2C_MSR*n* field descriptions (Continued)

Field	Description
RXAK	Receive Acknowledge. SDA value during the bus cycle acknowledge bit. If bit is low, it indicates an acknowledge signal was received after completion of 8 bits of data transmission on the bus. If bit is high, it means no acknowledge signal is detected at the 9th clock. 0 Acknowledge received. 1 No acknowledge received.

19.3.1.5 I²C Data I/O Register (I2C_MDR*n*)

Address: Base + 0x10 (I2C_MDR1)
Base + 0x30 (I2C_MDR2)
Base + 0x50 (I2C_MDR3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-13. I²C Data I/O Register (I2C_MDR*n*)

Table 19-9. I2C_MDR*n* field descriptions

Field	Description
D[7:0][1]	Master Transmit Mode. When data is written to this register, a data transfer is initiated. The most significant bit is sent first. **Note:** In this mode, the first data byte written to DR. Assertion of STA is used for the address transfer and should be comprised of the calling address (in position D[7]:D[1]) concatenated with the required R/$\overline{W}$ bit (in position D0). In Master Receive Mode, reading this register initiates next byte data receiving. In Slave Mode, the same functions are available after an address match occurs.

[1] When I²C is disabled (bitfield “EN” in I²C control register is equal to zero), this register always reads back zero, whatever value is written to it. The read out data of this register is updated only when the received data from the external device is changed, otherwise it remains unchanged.

19.3.1.6 I²C Interrupt Control Register (I2C_ICR)

Address: Base + 0x60 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	BNBE3	IE3	BNBE2	IE2	BNBE1	IE1	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-14. I²C Interrupt Control Register (I2C_ICR)

Table 19-10. I2C_ICR field descriptions

Field	Description
BNBE3	Bus Not Busy Enable 3. This bit lets module 3 generate an interrupt when the bus is not busy. BNBE3 indicates an idle condition. To clear the interrupt, software must write 0 to the bit position. Reset disables BNBE3.
IE3	Interrupt Enable 3, This bit routes the interrupt for module 3 to the CPU. Clear by writing 0 to this bit. Reset enables IE3.
BNBE2	Bus Not Busy Enable 2, This bit lets module 2 generate an interrupt when the bus is not busy. BNBE2 indicates an idle condition. To clear the interrupt, software must write 0 to the bit position. Reset disables BNBE2.
IE2	Interrupt Enable 2, This bit routes the interrupt for module 2 to the CPU. Clear by writing 0 to this bit. Reset enables IE2.
BNBE1	Bus Not Busy Enable 1, This bit lets module 1 generate an interrupt when the bus is not busy. BNBE1 indicates an idle condition. To clear the interrupt, software must write 0 to the bit position. Reset disables this bit.
IE1	Interrupt Enable 1, This bit routes the interrupt for module 1 to the CPU. Clear by writing 0 to this bit. Reset enables IE1.

The interrupt control register is common to three MPC5125 I²C modules. Each module generates an internal interrupt that can be routed to the CPU if the respective IE bit is set to 1.

Reset condition is IE set, and all other enable bits clear.

The BNBE bit lets the module generate an interrupt when the bus becomes not-busy. This implies receipt of a STOP condition, for which the module normally does not generate an interrupt. Because bus-not-busy is an idle condition, it is necessary for software responding to this interrupt to clear the BNBE bit to clear the interrupt condition. Otherwise, the interrupt condition persists until another I²C transaction is initiated.

19.3.1.7 I²C Filter Register (I2C_MIFR)

Address: Base + 0x64 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	FR3	FR2	FR1	FR0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-15. I²C Filter Register (I2C_MIFR)

Table 19-11. I2C_MIFR field descriptions

Field	Description
FR[7:4]	Bits 7 to 4 contain the programming controls for the width of glitch (in terms of IPS clock cycles) the filter should absorb; in other words, the filter does not let glitches less than or equal to this width setting pass. 0000 No Filter/Bypass 0001 Filter glitches as wide as 1 IPS clock cycle. 0010 Filter glitches as wide as 2 IPS clock cycles. 0011 Filter glitches as wide as 3 IPS clock cycles. 0100 Filter glitches as wide as 4 IPS clock cycles. 0101 Filter glitches as wide as 5 IPS clock cycles. 0110 Filter glitches as wide as 6 IPS clock cycles. 0111 Filter glitches as wide as 7 IPS clock cycles. 1000 Filter glitches as wide as 8 IPS clock cycles. 1001 Filter glitches as wide as 9 IPS clock cycles . 1010 Filter glitches as wide as 10 IPS clock cycles . 1011 Filter glitches as wide as 11 IPS clock cycles. 1100 Filter glitches as wide as 12 IPS clock cycles. 1101 Filter glitches as wide as 13 IPS clock cycles. 1110 Filter glitches as wide as 14 IPS clock cycles. 1111 Filter glitches as wide as 15 IPS clock cycles.

This filter can absorb glitches on the I²C clock and data lines for each I²C module. The width of the glitch to absorb is specified in terms of number of IPS clock cycles. This glitch filter control register is provided for all three I²C modules.

The programming of the glitch filter is simple; the programmer only needs to specify the size of glitch (in terms of IPS cycles) for the filter to absorb and not pass.

19.4 Initialization Sequence

Reset puts the I²C control register to its default status. Before the interface can transfer serial data, the following initialization procedure must be done:

Update the frequency divider register and select the required division ratio to obtain the SCL frequency from the IPS clock.

1. Calculate the divider according to the ips clock frequency and the expected SCL frequency: scl_divider = f_IPS/f_SCL.
2. Look in Table 19-6 to find out value of the Frequency Divider Bit (FDR) in the I2C_MFDR*n* register according to the calculated divider.
3. Set the prescaler in frequency divider register equal to FDR obtain from look-up table.

Update the I^2C address register to define a slave address.

Set the control register EN bit to enable the I^2C interface system.

Modify the control register bits to select master/slave mode, transmit/receive mode, and interrupt enable or not.

19.5 Transfer Initiation and Interrupt

In master transmit mode, a data transfer is initiated when data is written to the DATA register. The most significant bit is sent first.

In master receive mode, reading this register initiates next byte data receiving.

In slave mode, the same functions are available after an address match occurs. Data transfer is initiated by:

- Writing to the DATA register for slave transmits

or

- A dummy reading from the DATA register in slave receive mode occurs.

The I^2C interrupt STATUS register bit is set when an interrupt is pending. If the CONTROL register interrupt enable bit is set, the I^2C interrupt STATUS register bit, if set, causes a processor interrupt request. The interrupt bit sets when one of the following events occurs:

- A complete 1-Byte transfer (set at falling edge of ninth clock) occurs.
- A receive calling address matches its own specific address in slave receive mode.
- Arbitration is lost.

19.5.1 Post-Transfer Software Response

In the interrupt service routine, software must clear the IF status bit first. The CF status bit is cleared automatically by reading from the data I/O register (I2C_MDR*n*) in receive mode or writing to I2C_MDR*n* in transmit mode.

Software may service the bus I/O in the main program by monitoring the IF status bit if the interrupt function is disabled. Polling should monitor the IF status bit rather than the CF bit because its operation is different when arbitration is lost.

When an interrupt occurs at the end of the address cycle, the master is always in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit in the DATA register, the TX control bit should be toggled at this stage.

During slave mode address cycles (AAS = 1), the SRW bit in the STATUS register is read to determine the direction of the subsequent transfer and the TX control bit is programmed accordingly. The SRW bit is not valid for data cycles (AAS = 0) when operating in slave mode. Therefore, the TX bit in the control register should be read to determine the direction of the current transfer.

19.5.2 Slave Mode

In the slave interrupt service routine, the AAS bit should be tested to determine if a calling of its own address was received. If AAS is set, software should set the Tx/Rx mode select bit (control register Tx bit) according to the R/$\overline{W}$ command bit (SRW). Writing to the CONTROL register automatically clears AAS. The slave interrupt service routine should also move the data, depending on whether it acts as a transmitter or a receiver, as follows:

- For a slave transmitter, the slave interrupt service routine must initiate a data transfer by writing information to the DATA register.
- For a slave receiver, the slave interrupt service routine must initiate a transfer by performing a dummy read from the DATA register. The slave drives SCL low between byte transfers. SCL is released when the DATA register is accessed in the required mode.

In slave transmitter routine, RXAK must be tested before transmitting the next data byte. Setting RXAK means an end of data signal from the master receiver. Then, software causes a switch from transmitter mode to receiver mode. A dummy read then releases the SCL line letting the master generate a STOP signal.

19.5.3 Special Note on AKF

When the I^2C module loses arbitration (AL) and is addressed as slave (AAS), the I^2C module generates two interrupt (IF) requests: one on the rising edge of the acknowledge clock pulse and one on the falling edge of the acknowledge clock pulse (the legacy time to do so). In this specific case, the interrupt (IF = 1) is really the second one set by the hardware.

The software programmer must use this AKF bit to distinguish the first interrupt request (non-legacy) from the second interrupt request (legacy) and may use the value of AKF (in addition to all the usual bits the software checks) to determine when to take action for the AL and AAS.

- First interrupt on rising edge of acknowledge cycle — software should not take AL and AAS action
- Second interrupt on falling edge of acknowledge — software should take AL and AAS action

The AKF bit is set only for the second interrupt and must be cleared by software writing it low in the interrupt routine.

Figure 19-16 on typical software flow for I^2C routines clearly illustrates how the AKF bit may be used.

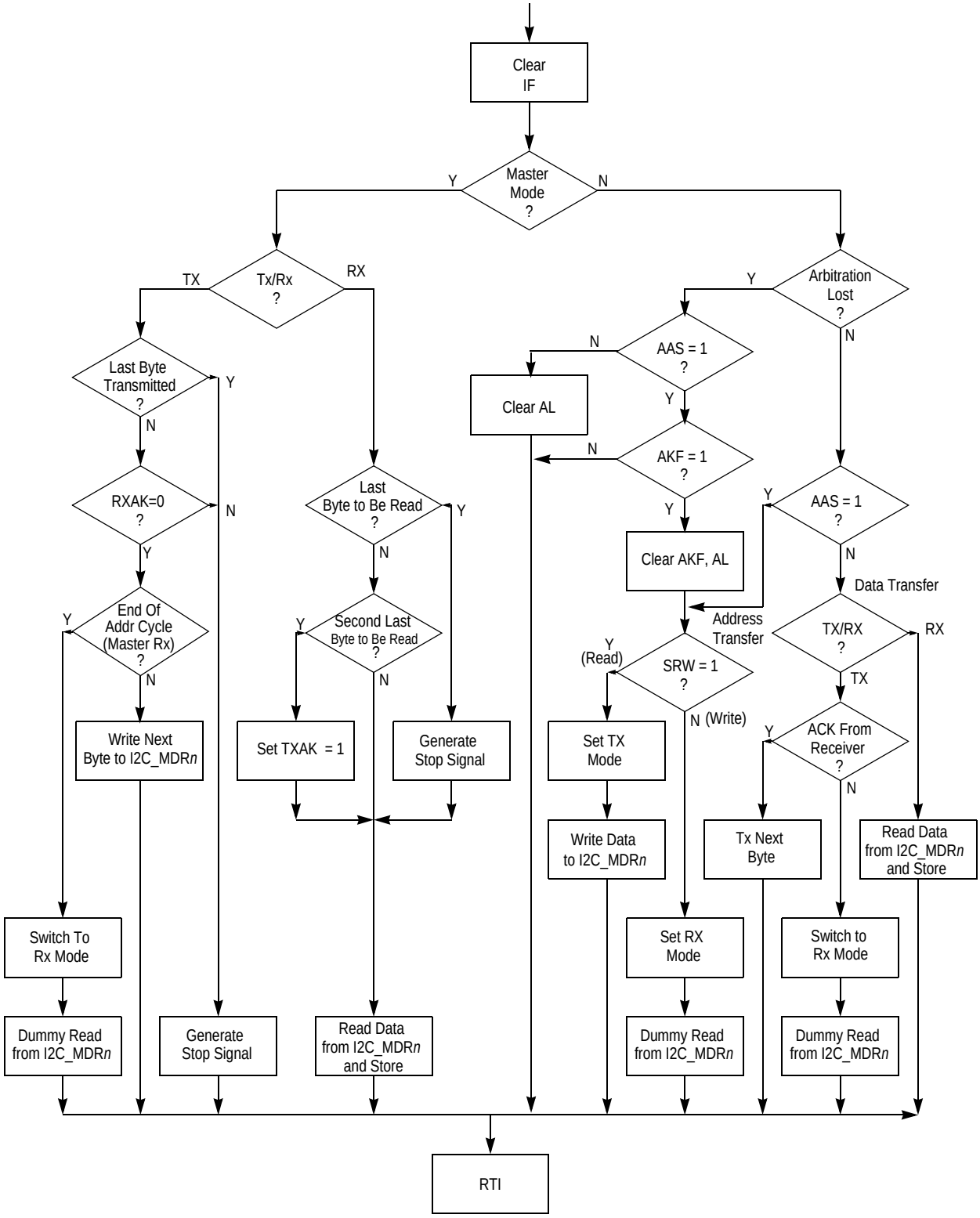


Figure 19-16. Software Flowchart of Typical I²C Interrupt Routine

This page is intentionally left blank.

Chapter 20
I/O Control

20.1 Introduction

20.1.1 Overview

The I/O control block controls the functional muxing and configuration of the pads. Configurable parameters include slew rate, Schmitt trigger input and pull-down/up, and a global Output Buffer Enable (OBE).

20.1.2 Features

- Functional pin muxing control
- Pad slew rate control
- Pad Schmitt trigger control
- Pad pull-down/pull-up control
- Global OBE control

20.2 Memory Map and Register Definition

20.2.1 Memory Map

Table 20-1. I/O Control Memory Map

Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Register[2]	Access	Reset Value	Section/Page
0x000	IO_CTL_MEM—MEM pad control register	R/W	0x00	20.2.2.1/20-502
0x001	IO_CTL_GBOBE—Global Output Enable Control Register	R/W	0x00	20.2.2.2/20-503
0x002–0x003	Reserved			
0x004	IO_CTL_LPC_CLK—LPC_CLK pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x005	IO_CTL_LPC_OE_B— PC_OE_B pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504

Table 20-1. I/O Control Memory Map (continued)

Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Register[2]	Access	Reset Value	Section/Page
0x006	IO_CTL_LPC_RWB—LPC_RWB pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x007	IO_CTL_LPC_CS0_B—LPC_CS0_B pad control register (STD_PU)	R/W	0x1B	20.2.2.3.1/20-504
0x008	IO_CTL_LPC_ACK_B—LPC_ACK_B pad control register (STD_PU)	R/W	0x1B	20.2.2.3.1/20-504
0x009	IO_CTL_LPC_AX03—LPC_AX03 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x00A	IO_CTL_EMB_AX02—EMB_AX02 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x00B	IO_CTL_EMB_AX01—EMB_AX01 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x00C	IO_CTL_EMB_AX00—EMB_AX00 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x00D	IO_CTL_EMB_AD31—EMB_AD31 pad control register (STD_PU_ST)	R/W	0x03	20.2.2.3.2/20-505
0x00E	IO_CTL_EMB_AD30—EMB_AD30 pad control register (STD_PU_ST)	R/W	0x03	20.2.2.3.2/20-505
0x00F	IO_CTL_EMB_AD29—EMB_AD29 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x010	IO_CTL_EMB_AD28—EMB_AD28 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x011	IO_CTL_EMB_AD27—EMB_AD27 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x012	IO_CTL_EMB_AD26—EMB_AD26 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x013	IO_CTL_EMB_AD25—EMB_AD25 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x014	IO_CTL_EMB_AD24—EMB_AD24 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x015	IO_CTL_EMB_AD23—EMB_AD23 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x016	IO_CTL_EMB_AD22—EMB_AD22 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x017	IO_CTL_EMB_AD21—EMB_AD21 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x018	IO_CTL_EMB_AD20—EMB_AD20 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504

Table 20-1. I/O Control Memory Map (continued)

Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Register[2]	Access	Reset Value	Section/Page
0x019	IO_CTL_EMB_AD19—EMB_AD19 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x01A	IO_CTL_EMB_AD18—EMB_AD18 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x01B	IO_CTL_EMB_AD17—EMB_AD17 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x01C	IO_CTL_EMB_AD16—EMB_AD16 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x01D	IO_CTL_EMB_AD15—EMB_AD15 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x01E	IO_CTL_EMB_AD14—EMB_AD14 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x01R	IO_CTL_EMB_AD13—EMB_AD13 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x020	IO_CTL_EMB_AD12—EMB_AD12 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x021	IO_CTL_EMB_AD11—EMB_AD11 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x022	IO_CTL_EMB_AD10—EMB_AD10 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x023	IO_CTL_EMB_AD09—EMB_AD09 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x024	IO_CTL_EMB_AD08—EMB_AD08 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x025	IO_CTL_EMB_AD07—EMB_AD07 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x026	IO_CTL_EMB_AD06—EMB_AD06 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x027	IO_CTL_EMB_AD05—EMB_AD05 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x028	IO_CTL_EMB_AD04—EMB_AD04 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x029	IO_CTL_EMB_AD03—EMB_AD03 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x02A	IO_CTL_EMB_AD02—EMB_AD02 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x02B	IO_CTL_EMB_AD01—EMB_AD01 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504

Table 20-1. I/O Control Memory Map (continued)

Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Register[2]	Access	Reset Value	Section/Page
0x02C	IO_CTL_EMB_AD00—EMB_AD00 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-504
0x02D	IO_CTL_NFC_CE0_B—NFC_CE0_B pad control register (STD_PU)	R/W	0x1B	20.2.2.3.1/20-504
0x02E	IO_CTL_NFC_RB—NFC_RB pad control register (STD_PU_ST)	R/W	0x07	20.2.2.3.2/20-505
0x02F	IO_CTL_DIU_CLK—DIU_CLK pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x030	IO_CTL_DIU_DE—DIU_DE pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x031	IO_CTL_DIU_HSYNC—DIU_HSYNC pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x032	IO_CTL_DIU_VSYNC—DIU_VSYNC pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x033	IO_CTL_DIU_LD00—DIU_LD00 pad control register (STD_PU_ST)	R/W	0x1C	20.2.2.3.2/20-505
0x034	IO_CTL_DIU_LD01—DIU_LD01 pad control register (STD_PU)	R/W	0x18	20.2.2.3.1/20-504
0x035	IO_CTL_DIU_LD02—DIU_LD02 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x036	IO_CTL_DIU_LD03—DIU_LD03 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x037	IO_CTL_DIU_LD04—DIU_LD04 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x038	IO_CTL_DIU_LD05—DIU_LD05 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x039	IO_CTL_DIU_LD06—DIU_LD06 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x03A	IO_CTL_DIU_LD07—DIU_LD07 pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-505
0x03B	IO_CTL_DIU_LD08—DIU_LD08 pad control register (STD_PU_ST)	R/W	0x1C	20.2.2.3.2/20-505
0x03C	IO_CTL_DIU_LD09—DIU_LD09 pad control register (STD_PU)	R/W	0x18	20.2.2.3.1/20-504
0x03D	IO_CTL_DIU_LD10—DIU_LD10 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x03E	IO_CTL_DIU_LD11—DIU_LD11 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504

Table 20-1. I/O Control Memory Map (continued)

Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Register[2]	Access	Reset Value	Section/Page
0x03F	IO_CTL_DIU_LD12—DIU_LD12 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x040	IO_CTL_DIU_LD13—DIU_LD13 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x041	IO_CTL_DIU_LD14—DIU_LD14 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x042	IO_CTL_DIU_LD15—DIU_LD15 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x043	IO_CTL_DIU_LD16—DIU_LD16 pad control register (STD_PU_ST)	R/W	0x18	20.2.2.3.2/20-505
0x044	IO_CTL_DIU_LD17—DIU_LD17 pad control register (STD_PU_ST)	R/W	0x18	20.2.2.3.2/20-505
0x045	IO_CTL_DIU_LD18—DIU_LD18 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x046	IO_CTL_DIU_LD19—DIU_LD19 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x047	IO_CTL_DIU_LD20—DIU_LD20 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x048	IO_CTL_DIU_LD21—DIU_LD21 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x049	IO_CTL_DIU_LD22—DIU_LD22 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x04A	IO_CTL_DIU_LD23—DIU_LD23 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x04B	IO_CTL_I2C2_SCL—I2C2_SCL pad control register (STD_PU_ST)	R/W	0x1C	20.2.2.3.2/20-505
0x04C	IO_CTL_I2C2_SDA—I2C2_SDA pad control register (STD_PU_ST)	R/W	0x1C	20.2.2.3.2/20-505
0x04D	IO_CTL_CAN1_TX—CAN1_TX pad control register (STD_PU_ST)	R/W	0x18	20.2.2.3.2/20-505
0x04E	IO_CTL_CAN2_TX—CAN2_TX pad control register (STD_PU_ST)	R/W	0x18	20.2.2.3.2/20-505
0x04F	IO_CTL_I2C1_SCL—I2C1_SCL pad control register (STD_PU_ST)	R/W	0x1C	20.2.2.3.2/20-505
0x050	IO_CTL_I2C1_SDA—I2C1_SDA pad control register (STD_PU_ST)	R/W	0x1C	20.2.2.3.2/20-505
0x051	IO_CTL_FEC1_TXD_2—FEC1_TXD_2 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504

Table 20-1. I/O Control Memory Map (continued)

Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Register[2]	Access	Reset Value	Section/Page
0x052	IO_CTL_FEC1_TXD_3—FEC1_TXD_3 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x053	IO_CTL_FEC1_RXD_2—FEC1_RXD_2 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x054	IO_CTL_FEC1_RXD_3—FEC1_RXD_3 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x055	IO_CTL_FEC1_CRS—FEC1_CRS pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x056	IO_CTL_FEC1_TX_ER—FEC1_TX_ER pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x057	IO_CTL_FEC1_RXD_1—FEC1_RXD_1 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x058	IO_CTL_FEC1_TXD_1—FEC1_TXD_1 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x059	IO_CTL_FEC1_MDC—FEC1_MDC pad control register (STD_PU)	R/W	0x04	20.2.2.3.1/20-504
0x05A	IO_CTL_FEC1_RX_ER—FEC1_RX_ER pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x05B	IO_CTL_FEC1_MDIO—FEC1_MDIO pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-505
0x05C	IO_CTL_FEC1_RXD_0—FEC1_RXD_0 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x05D	IO_CTL_FEC1_TXD_0—FEC1_TXD_0 pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-505
0x05E	IO_CTL_FEC1_TX_CLK—FEC1_TX_CLK pad control register (STD_PU_ST)	R/W	0x04	20.2.2.3.2/20-505
0x05F	IO_CTL_FEC1_RX_CLK—FEC1_RX_CLK pad control register (STD_PU_ST)	R/W	0x04	20.2.2.3.2/20-505
0x060	IO_CTL_FEC1_RX_DV—FEC1_RX_DV pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-505
0x061	IO_CTL_FEC1_TX_EN—FEC1_TX_EN pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x062	IO_CTL_FEC1_COL—FEC1_COL pad control register (STD_PU_ST)	R/W	0x04	20.2.2.3.2/20-505
0x063	IO_CTL_USB1_DATA0—USB1_DATA0 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x064	IO_CTL_USB1_DATA1—USB1_DATA1 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504

Table 20-1. I/O Control Memory Map (continued)

Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Register[2]	Access	Reset Value	Section/Page
0x065	IO_CTL_USB1_DATA2—USB1_DATA2 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x066	IO_CTL_USB1_DATA3—USB1_DATA3 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x067	IO_CTL_USB1_DATA4—USB1_DATA4 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x068	IO_CTL_USB1_DATA5—USB1_DATA5 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x069	IO_CTL_USB1_DATA6—USB1_DATA6 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x06A	IO_CTL_USB1_DATA7—USB1_DATA7 pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-505
0x06B	IO_CTL_USB1_STOP—USB1_STOP pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-505
0x06C	IO_CTL_USB1_CLK—USB1_CLK pad control register (STD_PU_ST)	R/W	0x04	20.2.2.3.2/20-505
0x06D	IO_CTL_USB1_NEXT—USB1_NEXT pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x06E	IO_CTL_USB1_DIR—USB1_DIR pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-505
0x06F	IO_CTL_SDHC1_CLK—SDHC1_CLK pad control register (STD_PU)	R/W	0x18	20.2.2.3.1/20-504
0x070	IO_CTL_SDHC1_CMD—SDHC1_CMD pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x071	IO_CTL_SDHC1_D0—SDHC1_D0 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x072	IO_CTL_SDHC1_D1—SDHC1_D1 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x073	IO_CTL_SDHC1_D2—SDHC1_D2 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x074	IO_CTL_SDHC1_D3—SDHC1_D3 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x075	IO_CTL_PSC_MCLK_IN—PSC_MCLK_IN pad control register (STD_PU_ST)	R/W	0x04	20.2.2.3.2/20-505
0x076	IO_CTL_PSC0_0—PSC0_0 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x077	IO_CTL_PSC0_1—PSC0_1 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504

Table 20-1. I/O Control Memory Map (continued)

Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Register[2]	Access	Reset Value	Section/Page
0x078	IO_CTL_PSC0_2—PSC0_2 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x079	IO_CTL_PSC0_3—PSC0_3 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x07A	IO_CTL_PSC0_4—PSC0_4 pad control register (STD_PU)	R/W	0x18	20.2.2.3.1/20-504
0x07B	IO_CTL_PSC1_0—PSC1_0 pad control register (STD_PU)	R/W	0x18	20.2.2.3.1/20-504
0x07C	IO_CTL_PSC1_1—PSC1_1 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x07D	IO_CTL_PSC1_2—PSC1_2 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x07E	IO_CTL_PSC1_3—PSC1_3 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-504
0x07F	IO_CTL_PSC1_4—PSC1_4 pad control register (STD_PU)	R/W	0x18	20.2.2.3.1/20-504
0x080	IO_CTL_J1850_TX—J1850_TX pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-505
0x081	IO_CTL_J1850_RX—J1850_RX pad control register (STD_PU_ST)	R/W	0x18	20.2.2.3.2/20-505
0x082–0xFFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 Pads using the STD_PU_ST register have a Schmitt trigger input; pads using the STD_PU register do not.

20.2.2 Register Descriptions

20.2.2.1 IO_CTL_MEM Register

Address: Base + 0x000 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	16BIT	CONT_DS			DATA_DS		
W								
Reset	0	0	0	0	0	0	0	0

Figure 20-1. MEM IO Control Register (IO_CTL_MEM)

Table 20-2. IO_CTL_MEM field descriptions

Field	Description
16BIT	Enables pin muxing in DRAM 16 bit mode. 0 All DRAM pads are used for DRAM functionality. 1 MDQ[31:16], MDM[3:2] and MDQS[3:2] have alternate GPIO/GPT functionality. See Table 3-1.
CONT_DS	CONT_DS controls the slew rate of all DRAM control pads (MCS, MA[15:0], MCK, $\overline{\text{MCK}}$, MODT, MBA[2:0], MCAS, MRAS, MCKE, and MWE). 000 DDR pad configuration 0 001 DDR pad configuration 1 010 DDR pad configuration 2 011 DDR pad configuration 3 100 Reserved 101 Reserved 110 DDR pad configuration 6 111 DDR pad configuration 7 **Note:** DDR pad configurations are defined in the *MPC5125 Microcontroller Data Sheet*.
DATA_DS	DATA_DS controls the slew rate of all DRAM data pads (MDQ[31:0], MDM[3:0] and MDQS[3:0]). 000 DDR pad configuration 0 001 DDR pad configuration 1 010 DDR pad configuration 2 011 DDR pad configuration 3 100 Reserved 101 Reserved 110 DDR pad configuration 6 111 DDR pad configuration 7 **Note:** The configured DDR data pads is also valid for the GPIO/GPT lines in 16BIT mode configuration. **Note:** DDR pad configurations are defined in the *MPC5125 Microcontroller Data Sheet*.

20.2.2.2 Global Output Enable Control Register (IO_CTL_GBOBE)

Address: Base + 0x001 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	GB_OBE
W								
Reset	0	0	0	0	0	0	0	0

Figure 20-2. Global output enable control register (IO_CTL_GBOBE)

Table 20-3. IO_CTL_GBOBE field descriptions

Field	Description
GB_OBE	Global Output Enable 0 All default output pads except Test and boot related pads are driven to High Z. 1 Normal work. This bit only controls function 0. It can be overridden by selecting a function other than function 0.

20.2.2.3 IO_CTL_PAD Registers Descriptions

There are six different types of IO_CTL_PAD registers. The drive strength of each pad and the functional muxing is programmable. The following different types are used:

- Standard with pull-up/down resistors (STD_PU)
  - Functional muxing
  - Programmable slew rate
  - Programmable pull-up/down resistors
- Standard with pull-up/down resistors and Schmitt trigger input (STD_PU_ST)
  - Functional muxing
  - Programmable slew rate
  - Programmable Schmitt trigger input
  - Programmable pull-up/down resistors

Figure 20-3 and Figure 20-4 describe the two different types of configuration registers. Table 20-6 shows which type of configuration register is responsible to configure the different pads. Additionally, it shows the different functional muxing possibilities.

20.2.2.3.1 Standard with Pull-up/down Resistors (STD_PU)

Address: See Table 20-6 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	FUNCMUX		PUD	PUE	0	DS	
W								
Reset	0	0	0	0	0	0	0	0

Figure 20-3. Standard with Pull-up/down Resistors (STD_PU)

Table 20-4. STD_PU field descriptions

Field	Description
FUNCMUX	Functional multiplexing. Controls the functional pin muxing of the pad. 00 ALT0 (default) 01 ALT1 10 ALT2 11 ALT3
PUD	Pull-up/down direction. Controls the direction of the pull resistors. 0 Pull-down resistor enabled, if PUE is 1. 1 Pull-up resistor enabled, if PUE is 1.
PUE	Pull-up/down enable. Enables the pull-up/down usage. 0 Pull resistor is disabled. 1 Pull resistor is enabled.

Table 20-4. STD_PU field descriptions (continued)

Field	Description
DS	Drive select /slew rate. Controls the drive select and slew rate of the general I/O pad. 00 General I/O slew rate configuration 0 01 General I/O slew rate configuration 1 10 General I/O slew rate configuration 2 11 General I/O slew rate configuration 3 **Note:** Slew rate classes are defined in the *MPC5125 Microcontroller Data Sheet*

20.2.2.3.2 Standard with Pull-up/down Resistors and Schmitt trigger input (STD_PU_ST)

Address: See Table 20-6 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	FUNCMUX		PUD	PUE	ST	DS	
W								
Reset	0	0	0	0	0	0	0	0

Figure 20-4. Standard with Pull-up/down Resistors and Schmitt trigger I Control Register (STD_PU_ST)

Table 20-5. STD_PU_ST field descriptions

Field	Description
FUNCMUX	Functional multiplexing. Controls the functional pin muxing of the pad. 00 ALT0 (default) 01 ALT1 10 ALT2 11 ALT3
PUD	Pull-up/down direction. Controls the direction of the pull resistors. 0 If pull-up/down is enabled (PUE = 1), pull-down resistor enabled. 1 If pull-up/down is enabled (PUE = 1), pull-up resistor enabled.
PUE	Pull-up/down enable. Enables the pull-up/down usage. 0 Pull-up/down resistor is disabled. 1 Pull-up/down resistor is enabled.
ST	Schmitt trigger. Enables the Schmitt trigger input of the pad. 0 Schmitt Trigger input is disabled. 1 Schmitt Trigger input is enabled.
DS	Drive select /slew rate. Controls the drive select and slew rate of the general I/O pad. 00 General IO slew rate configuration 0. 01 General IO slew rate configuration 1. 10 General IO slew rate configuration 2. 11 General IO slew rate configuration 3. **Note:** Slew rate classes are defined in the *MPC5125 Microcontroller Data Sheet*.

20.2.2.3.3 Pad I/O Control Register

Table 20-6. Pad I/O Control Register Table

PAD Name	Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Type	Reset Value [6:0]	FUNCMUX
LPC_CLK	0x04	STD_PU	0x03 7'b00_000_11[2]	00 LPC_CLK 01 TPA1 10 Reserved 11 GPIO04
$\overline{\text{LPC\_OE}}$	0x05	STD_PU	0x03 7'b00_000_11[2]	00 $\overline{\text{LPC\_OE}}$ 01 PSC3_3 10 Reserved 11 GPIO05
LPC_R/$\overline{\text{W}}$	0x06	STD_PU	0x03 7'b00_000_11[2]	00 LPC_R/$\overline{\text{W}}$ 01 PSC3_4 10 Reserved 11 GPIO06
$\overline{\text{LPC\_CS0}}$	0x07	STD_PU	0x1B 7'b00_110_11[3]	00 $\overline{\text{LPC\_CS0}}$ 01 Reserved 10 Reserved 11 GPIO07
$\overline{\text{LPC\_ACK}}$	0x08	STD_PU	0x1B 7'b00_110_11[3]	00 $\overline{\text{LPC\_ACK}}$/LPC_BURST 01 $\overline{\text{NFC\_CE1}}$ 10 $\overline{\text{LPC\_CS1}}$ 11 GPIO08
LPC_AX03	0x09	STD_PU	0x03 7'b00_000_11[4]	00 LPC_AX03/LPC_TS 01 $\overline{\text{NFC\_CE2}}$ 10 $\overline{\text{LPC\_CS2}}$ 11 Reserved
EMB_AX02	0x0A	STD_PU	0x03 7'b00_000_11[4]	00 LPC_AX02/LPC_TSIZ1 01 $\overline{\text{NFC\_CE3}}$ 10 $\overline{\text{LPC\_CS3}}$ 11 Reserved
EMB_AX01	0x0B	STD_PU	0x03 7'b00_000_11[4]	00 LPC_AX01/LPC_TSIZ0 01 Reserved 10 $\overline{\text{LPC\_CS4}}$ 11 Reserved
EMB_AX00	0x0C	STD_PU	0x03 7'b00_000_11[2]	00 LPC_AX00/LPC_ALE 01 Reserved 10 Reserved 11 Reserved
EMB_AD31	0x0D	STD_PU_ST	0x03 7'b00_000_11[2]	00 LPC_AD31/LPC_A16 01 PSC_MCLK_IN 10 Reserved 11 GPIO09
EMB_AD30	0x0E	STD_PU_ST	0x03 7'b00_000_11[2]	00 LPC_AD30/LPC_A15 01 CAN_CLK_IN 10 Reserved 11 GPIO10

Table 20-6. Pad I/O Control Register Table (continued)

PAD Name	Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Type	Reset Value [6:0]	FUNCMUX
EMB_AD29	0x0F	STD_PU	0x00 7'b00_000_00[5]	00 LPC_AD29/LPC_A14 01 Reserved 10 Reserved 11 GPIO11
EMB_AD28	0x10	STD_PU	0x00 7'b00_000_00[5]	00 LPC_AD28/LPC_A13 01 Reserved 10 Reserved 11 GPIO12
EMB_AD27	0x11	STD_PU	0x00 7'b00_000_00[5]	00 LPC_AD27/LPC_A12 01 Reserved 10 Reserved 11 GPIO13
EMB_AD26	0x12	STD_PU	0x00 7'b00_000_00[5]	00 LPC_AD26/LPC_A11 01 Reserved 10 Reserved 11 GPIO14
EMB_AD25	0x13	STD_PU	0x00 7'b00_000_00[5]	00 LPC_AD25/LPC_A10 01 Reserved 10 Reserved 11 GPIO15
EMB_AD24	0x14	STD_PU	0x00 7'b00_000_00[5]	00 LPC_AD24/LPC_A09 01 Reserved 10 Reserved 11 GPIO16
EMB_AD23	0x15	STD_PU	0x00 7'b00_000_00[5]	00 LPC_AD23/LPC_A08 01 Reserved 10 Reserved 11 GPIO17
EMB_AD22	0x16	STD_PU	0x00 7'b00_000_00[5]	00 LPC_AD22/LPC_A07 01 Reserved 10 Reserved 11 GPIO18
EMB_AD21	0x17	STD_PU	0x00 7'b00_000_00[5]	00 LPC_AD21/LPC_A06 01 Reserved 10 Reserved 11 GPIO19
EMB_AD20	0x18	STD_PU	0x00 7'b00_000_00[5]	00 LPC_AD20/LPC_A05 01 Reserved 10 Reserved 11 GPIO20
EMB_AD19	0x19	STD_PU	0x03 7'b00_000_11	00 LPC_AD19/LPC_A04/NFC_ALE 01 Reserved 10 RST_CONF_LPCWA 11 Reserved

Table 20-6. Pad I/O Control Register Table (continued)

PAD Name	Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Type	Reset Value [6:0]	FUNCMUX
EMB_AD18	0x1a	STD_PU	0x03 7'b00_000_11	00 LPC_AD18/LPC_A03/NFC_CLE 01 Reserved 10 RST_CONF_LPCMX 11 Reserved
EMB_AD17	0x1B	STD_PU	0x03 7'b00_000_11	00 LPC_AD17/LPC_A02/$\overline{\text{NFC\_RE}}$ 01 Reserved 10 RST_CONF_PLL_LOCK 11 Reserved
EMB_AD16	0x1C	STD_PU	0x03 7'b00_000_11	00 LPC_AD16/LPC_A01/$\overline{\text{NFC\_WE}}$ 01 Reserved 10 Reserved 11 Reserved
EMB_AD15	0x1D	STD_PU	0x03 7'b00_000_11	00 LPC_AD15/NFC_AD15 01 PSC2_0 10 RST_CONF_SYSOSCEN 11 GPIO21
EMB_AD14	0x1E	STD_PU	0x03 7'b00_000_11	00 LPC_AD14/NFC_AD14 01 PSC2_1 10 RST_CONF_PREDIV2 11 GPIO22
EMB_AD13	0x1F	STD_PU	0x03 7'b00_000_11	00 LPC_AD13/NFC_AD13 01 PSC2_2 10 RST_CONF_PREDIV1 11 GPIO23
EMB_AD12	0x20	STD_PU	0x03 7'b00_000_11	00 LPC_AD12/NFC_AD12 01 PSC2_3 10 RST_CONF_PREDIV0 11 GPIO24
EMB_AD11	0x21	STD_PU	0x03 7'b00_000_11	00 LPC_AD11/NFC_AD11 01 PSC2_4 10 RST_CONF_SPMF3 11 GPIO25
EMB_AD10	0x22	STD_PU	0x03 7'b00_000_11	00 LPC_AD10/NFC_AD10 01 PSC3_0 10 RST_CONF_SPMF2 11 GPIO26
EMB_AD09	0x23	STD_PU	0x03 7'b00_000_11	00 LPC_AD09/NFC_AD09 01 PSC3_1 10 RST_CONF_SPMF1 11 GPIO27
EMB_AD08	0x24	STD_PU	0x03 7'b00_000_11	00 LPC_AD08/NFC_AD08 01 PSC3_2 10 RST_CONF_SPMF0 11 GPIO28

Table 20-6. Pad I/O Control Register Table (continued)

PAD Name	Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Type	Reset Value [6:0]	FUNCMUX
EMB_AD07	0x25	STD_PU	0x03 7'b00_000_11	00 LPC_AD07/NFC_AD07 01 Reserved 10 RST_CONF_COREPLL4 11 Reserved
EMB_AD06	0x26	STD_PU	0x03 7'b00_000_11	00 LPC_AD06/NFC_AD06 01 Reserved 10 RST_CONF_COREPLL5 11 Reserved
EMB_AD05	0x27	STD_PU	0x03 7'b00_000_11	00 LPC_AD05/NFC_AD05 01 Reserved 10 RST_CONF_COREPLL6 11 Reserved
EMB_AD04	0x28	STD_PU	0x03 7'b00_000_11	00 LPC_AD04/NFC_AD04 01 Reserved 10 RST_CONF_LPCDBW1 11 Reserved
EMB_AD03	0x29	STD_PU	0x03 7'b00_000_11	00 LPC_AD03/NFC_AD03 01 Reserved 10 RST_CONF_LPCDBW0 11 Reserved
EMB_AD02	0x2A	STD_PU	0x03 7'b00_000_11	00 LPC_AD02/NFC_AD02 01 Reserved 10 RST_CONF_BMS 11 Reserved
EMB_AD01	0x2B	STD_PU	0x03 7'b00_000_11	00 LPC_AD01/NFC_AD01 01 Reserved 10 RST_CONF_LOC1 11 Reserved
EMB_AD00	0x2C	STD_PU	0x03 7'b00_000_11	00 LPC_AD00/NFC_AD00 01 Reserved 10 RST_CONF_LOC0 11 Reserved
$\overline{\text{NFC\_CE0}}$	0x2D	STD_PU	0x1B 7'b00_110_11	00 $\overline{\text{NFC\_CE0}}$ 01 Reserved 10 Reserved 11 GPIO29
NFC_$\overline{\text{RB}}$	0x2E	STD_PU_ST	0x07 7'b00_001_11	00 NFC_$\overline{\text{RB}}$ 01 Reserved 10 Reserved 11 GPIO30
DIU_CLK	0x2F	STD_PU	0x00 7'b00_000_00[6]	00 DIU_CLK 01 PSC4_0 10 USB1_DATA0 11 LPC_AX04

Table 20-6. Pad I/O Control Register Table (continued)

PAD Name	Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Type	Reset Value [6:0]	FUNCMUX
DIU_DE	0x30	STD_PU	0x00 7'b00_000_00[6]	00 DIU_DE 01 PSC4_1 10 USB1_DATA1 11 LPC_AX05
DIU_HSYNC	0x31	STD_PU	0x00 7'b00_000_00[6]	00 DIU_HSYNC 01 PSC4_2 10 USB1_DATA2 11 LPC_AX06
DIU_VSYNC	0x32	STD_PU	0x00 7'b00_000_00	00 DIU_VSYNC 01 PSC4_3 10 USB1_DATA3 11 GPIO31
DIU_LD00	0x33	STD_PU_ST	0x1C 7'b00_111_00	00 CAN3_RX 01 CLK_OUT2 10 DIU_LD00 11 GPIO32
DIU_LD01	0x34	STD_PU	0x18 7'b00_110_00	00 CAN3_TX 01 CLK_OUT3 10 DIU_LD01 11 GPIO33
DIU_LD02	0x35	STD_PU	0x00 7'b00_000_00[6]	00 DIU_LD02 01 PSC4_4 10 USB1_DATA4 11 LPC_AX07
DIU_LD03	0x36	STD_PU	0x00 7'b00_000_00[6]	00 DIU_LD03 01 PSC5_0 10 USB1_DATA5 11 LPC_AX08
DIU_LD04	0x37	STD_PU	0x00 7'b00_000_00[6]	00 DIU_LD04 01 PSC5_1 10 USB1_DATA6 11 LPC_AX09
DIU_LD05	0x38	STD_PU	0x00 7'b00_000_00	00 DIU_LD05 01 PSC5_2 10 USB1_DATA7 11 GPIO34
DIU_LD06	0x39	STD_PU	0x00 7'b00_000_00	00 DIU_LD06 01 PSC5_3 10 USB1_STOP 11 GPIO35
DIU_LD07	0x3A	STD_PU_ST	0x00 7'b00_000_00	00 DIU_LD07 01 PSC5_4 10 USB1_CLK 11 GPIO36

Table 20-6. Pad I/O Control Register Table (continued)

PAD Name	Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Type	Reset Value [6:0]	FUNCMUX
DIU_LD08	0x3B	STD_PU_ST	0x1C 7'b00_111_00	00 CAN4_RX 01 PSC6_0 10 DIU_LD08 11 GPIO37
DIU_LD09	0x3C	STD_PU	0x18 7'b00_110_00	00 CAN4_TX 01 PSC6_1 10 DIU_LD09 11 GPIO38
DIU_LD10	0x3D	STD_PU	0x00 7'b00_000_00	00 DIU_LD10 01 PSC6_2 10 USB1_NEXT 11 GPIO39
DIU_LD11	0x3E	STD_PU	0x00 7'b00_000_00	00 DIU_LD11 01 PSC6_3 10 USB1_DIR 11 GPIO40
DIU_LD12	0x3F	STD_PU	0x00 7'b00_000_00	00 DIU_LD12 01 PSC6_4 10 USB2_DATA0 11 GPT8
DIU_LD13	0x40	STD_PU	0x00 7'b00_000_00	00 DIU_LD13 01 PSC7_0 10 USB2_DATA1 11 GPT9
DIU_LD14	0x41	STD_PU	0x00 7'b00_000_00	00 DIU_LD14 01 PSC7_1 10 USB2_DATA2 11 GPT10
DIU_LD15	0x42	STD_PU	0x00 7'b00_000_00	00 DIU_LD15 01 PSC7_2 10 USB2_DATA3 11 GPT11
DIU_LD16	0x43	STD_PU_ST	0x18 7'b00_110_00	00 CLK_OUT0 01 I2C3_SCL 10 DIU_LD16 11 GPIO41
DIU_LD17	0x44	STD_PU_ST	0x18 7'b00_110_00	00 CLK_OUT1 01 I2C3_SDA 10 DIU_LD17 11 GPIO42
DIU_LD18	0x45	STD_PU	0x00 7'b00_000_00	00 DIU_LD18 01 PSC7_3 10 USB2_DATA4 11 GPT12

Table 20-6. Pad I/O Control Register Table (continued)

PAD Name	Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Type	Reset Value [6:0]	FUNCMUX
DIU_LD19	0x46	STD_PU	0x00 7'b00_000_00	00 DIU_LD19 01 PSC7_4 10 USB2_DATA5 11 GPT13
DIU_LD20	0x47	STD_PU	0x00 7'b00_000_00	00 DIU_LD20 01 PSC8_0 10 USB2_DATA6 11 GPT14
DIU_LD21	0x48	STD_PU	0x00 7'b00_000_00	00 DIU_LD21 01 PSC8_1 10 USB2_DATA7 11 GPIO15
DIU_LD22	0x49	STD_PU	0x00 7'b00_000_00	00 DIU_LD22 01 PSC8_2 10 USB2_DIR 11 GPIO43
DIU_LD23	0x4A	STD_PU	0x00 7'b00_000_00	00 DIU_LD23 01 PSC8_3 10 USB2_NEXT 11 GPIO44
I2C2_SCL	0x4B	STD_PU_ST	0x1C 7'b00_111_00	00 I2C2_SCL 01 PSC8_4 10 USB2_CLK 11 GPIO45
I2C2_SDA	0x4C	STD_PU_ST	0x1C 7'b00_111_00	00 I2C2_SDA 01 PSC9_4 10 USB2_STOP 11 GPIO46
CAN1_TX	0x4D	STD_PU_ST	0x18 7'b00_110_00	00 CAN1_TX 01 PSC9_0 10 I2C2_SCL 11 GPIO47
CAN2_TX	0x4E	STD_PU_ST	0x18 7'b00_110_00	00 CAN2_TX 01 PSC9_1 10 I2C2_SDA 11 GPIO48
I2C1_SCL	0x4F	STD_PU_ST	0x1C 7'b00_111_00	00 I2C1_SCL 01 PSC9_2 10 CAN3_RX 11 GPIO49
I2C1_SDA	0x50	STD_PU_ST	0x1C 7'b00_111_00	00 I2C1_SDA 01 PSC9_3 10 CAN3_TX 11 GPIO50

Table 20-6. Pad I/O Control Register Table (continued)

PAD Name	Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Type	Reset Value [6:0]	FUNCMUX
FEC1_TXD_2	0x51	STD_PU	0x00 7'b00_000_00	00 FEC1_TXD_2 01 PSC2_0 10 USB2_DATA0 11 GPIO51
FEC1_TXD_3	0x52	STD_PU	0x00 7'b00_000_00	00 FEC1_TXD_3 01 PSC2_1 10 USB2_DATA1 11 GPIO52
FEC1_RXD_2	0x53	STD_PU	0x00 7'b00_000_00	00 FEC1_RXD_2 01 PSC2_2 10 USB2_DATA2 11 GPIO53
FEC1_RXD_3	0x54	STD_PU	0x00 7'b00_000_00	00 FEC1_RXD_3 01 PSC2_3 10 USB2_DATA3 11 GPIO54
FEC1_CRS	0x55	STD_PU	0x00 7'b00_000_00	00 FEC1_CRS 01 PSC2_4 10 USB2_DATA4 11 GPIO55
FEC1_TX_ER	0x56	STD_PU	0x00 7'b00_000_00	00 FEC1_TX_ER 01 PSC3_0 10 USB2_DATA5 11 GPIO56
FEC1_RXD_1	0x57	STD_PU	0x00 7'b00_000_00	00 FEC1_RXD_1/RMII_RX1 01 PSC3_1 10 USB2_DATA6 11 GPIO57
FEC1_TXD_1	0x58	STD_PU	0x00 7'b00_000_00	00 FEC1_TXD_1/RMII_TX1 01 PSC3_2 10 USB2_DATA7 11 GPIO58
FEC1_MDC	0x59	STD_PU	0x04 7'b00_001_00	00 FEC1_MDC 01 PSC3_3 10 USB2_DIR 11 GPIO59
FEC1_RX_ER	0x5A	STD_PU	0x00 7'b00_000_00	00 FEC1_RX_ER/RMII_RX_ER 01 PSC3_4 10 USB2_NEXT 11 GPIO60
FEC1_MDIO	0x5B	STD_PU_ST	0x00 7'b00_000_00	00 FEC1_MDIO 01 Reserved 10 USB2_CLK 11 GPIO61

Table 20-6. Pad I/O Control Register Table (continued)

PAD Name	Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Type	Reset Value [6:0]	FUNCMUX
FEC1_RXD_0	0x5C	STD_PU	0x00 7'b00_000_00	00 FEC1_RXD_0/RMII_RX0 01 Reserved 10 USB2_STOP 11 GPIO62
FEC1_TXD_0	0x5D	STD_PU_ST	0x00 7'b00_000_00	00 FEC1_TXD_0/RMII_TX0 01 Reserved 10 NFC_RB1 11 GPIO63
FEC1_TX_CLK	0x5E	STD_PU_ST	0x04 7'b00_001_00	00 FEC1_TX_CLK/RMII_REF_CLK 01 PSC0_0 10 Reserved 11 GPIO04
FEC1_RX_CLK	0x5F	STD_PU_ST	0x04 7'b00_001_00	00 FEC1_RX_CLK 01 PSC0_1 10 NFC_RB2 11 GPIO05
FEC1_RX_DV	0x60	STD_PU_ST	0x00 7'b00_000_00	00 FEC1_RX_DV/RMII_CRS_DV 01 PSC0_2 10 NFC_RB3 11 GPIO06
FEC1_TX_EN	0x61	STD_PU	0x00 7'b00_000_00	00 FEC1_TX_EN/RMII_TX_EN 01 PSC0_3 10 Reserved 11 GPIO07
FEC1_COL	0x62	STD_PU_ST	0x04 7'b00_001_00	00 FEC1_COL 01 PSC0_4 10 Reserved 11 GPIO08
USB1_DATA0	0x63	STD_PU	0x00 7'b00_000_00[7]	00 USB1_DATA0 01 PSC1_0 10 FEC2_RXD_1/RMII_RX1 11 Reserved
USB1_DATA1	0x64	STD_PU	0x00 7'b00_000_00[7]	00 USB1_DATA1 01 PSC1_1 10 FEC2_TXD_1/RMII_TX1 11 Reserved
USB1_DATA2	0x65	STD_PU	0x00 7'b00_000_00[7]	00 USB1_DATA2 01 PSC1_2 10 FEC2_MDC 11 Reserved
USB1_DATA3	0x66	STD_PU	0x00 7'b00_000_00[7]	00 USB1_DATA3 01 PSC1_3 10 FEC2_RX_ER/RMII_RX_ER 11 Reserved

Table 20-6. Pad I/O Control Register Table (continued)

PAD Name	Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Type	Reset Value [6:0]	FUNCMUX
USB1_DATA4	0x67	STD_PU	0x00 7'b00_000_00[7]	00 USB1_DATA4 01 PSC1_4 10 FEC2_MDIO/RMII_MDIO 11 Reserved
USB1_DATA5	0x68	STD_PU	0x00 7'b00_000_00[7]	00 USB1_DATA5 01 PSC4_0 10 FEC2_RXD_0/RMII_RX0 11 Reserved
USB1_DATA6	0x69	STD_PU	0x00 7'b00_000_00[7]	00 USB1_DATA6 01 PSC4_1 10 FEC2_TXD_0/RMII_TX0 11 Reserved
USB1_DATA7	0x6A	STD_PU_ST	0x00 7'b00_000_00[7]	00 USB1_DATA7 01 PSC4_2 10 FEC2_TX_CLK/RMII_REF_CLK 11 Reserved
USB1_STOP	0x6B	STD_PU_ST	0x00 7'b00_000_00[7]	00 USB1_STOP 01 PSC4_3 10 FEC2_RX_CLK 11 Reserved
USB1_CLK	0x6C	STD_PU_ST	0x04 7'b00_001_00[7]	00 USB1_CLK 01 PSC4_4 10 FEC2_RX_DV/RMII_CRS_DV 11 Reserved
USB1_NEXT	0x6D	STD_PU	0x00 7'b00_000_00	00 USB1_NEXT 01 Reserved 10 FEC2_TX_EN/RMII_TX_EN 11 GPIO09
USB1_DIR	0x6E	STD_PU_ST	0x00 7'b00_000_00	00 USB1_DIR 01 Reserved 10 FEC2_COL 11 GPIO10
SDHC1_CLK	0x6F	STD_PU	0x18 7'b00_110_00	00 SDHC1_CLK 01 NFC_CE1 10 FEC2_TXD_2 11 GPIO11
SDHC1_CMD	0x70	STD_PU	0x00 7'b00_000_00	00 SDHC1_CMD 01 PSC5_0 10 FEC2_TXD_3 11 GPIO12
SDHC1_D0	0x71	STD_PU	0x00 7'b00_000_00	00 SDHC1_D0 01 PSC5_1 10 FEC2_RXD_2 11 GPIO13

Table 20-6. Pad I/O Control Register Table (continued)

PAD Name	Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Type	Reset Value [6:0]	FUNCMUX
SDHC1_D1	0x72	STD_PU	0x00 7'b00_000_00	00 SDHC1_D1_IRQ 01 PSC5_2 10 FEC2_RXD_3 11 LPC_CS5
SDHC1_D2	0x73	STD_PU	0x00 7'b00_000_00	00 SDHC1_D2 01 PSC5_3 10 FEC2_CRS 11 LPC_CS6
SDHC1_D3	0x74	STD_PU	0x00 7'b00_000_00	00 SDHC1_D3_CD 01 PSC5_4 10 FEC2_TX_ER 11 LPC_CS7
PSC_MCLK_IN	0x75	STD_PU_ST	0x04 7'b00_001_00	00 PSC_MCLK_IN 01 Reserved 10 Reserved 11 GPIO14
PSC0_0	0x76	STD_PU	0x00 7'b00_000_00	00 PSC0_0 01 SDHC2_CMD 10 GPT0 11 GPIO15
PSC0_1	0x77	STD_PU	0x00 7'b00_000_00	00 PSC0_1 01 SDHC2_D0 10 GPT1 11 GPIO16
PSC0_2	0x78	STD_PU	0x00 7'b00_000_00	00 PSC0_2 01 SDHC2_D1_IRQ 10 GPT2 11 GPIO17
PSC0_3	0x79	STD_PU	0x00 7'b00_000_00	00 PSC0_3 01 SDHC2_D2 10 GPT3 11 GPIO18
PSC0_4	0x7A	STD_PU	0x18 7'b00_110_00	00 PSC0_4 01 SDHC2_D3_CD 10 GPT4 11 CAN1_TX
PSC1_0	0x7B	STD_PU	0x18 7'b00_110_00	00 PSC1_0 01 SDHC2_CLK 10 GPT5 11 CAN2_TX
PSC1_1	0x7C	STD_PU	0x00 7'b00_000_00	00 PSC1_1 01 CAN_CLK_IN 10 GPT6 11 IRQ0

Table 20-6. Pad I/O Control Register Table (continued)

PAD Name	Offset from IOCONTROL_BASE (0xFF40_A000)[1]	Type	Reset Value [6:0]	FUNCMUX
PSC1_2	0x7D	STD_PU	0x00 7'b00_000_00	00 PSC1_2 01 TPA2 10 GPT7 11 $\overline{\text{IRQ1}}$
PSC1_3	0x7E	STD_PU	0x00 7'b00_000_00	00 PSC1_3 01 CKSTP_IN 10 NFC_$\overline{\text{RB}}$2 11 GPIO19
PSC1_4	0x7F	STD_PU	0x18 7'b00_110_00	00 PSC1_4 01 $\overline{\text{CKSTP\_OUT}}$ 10 $\overline{\text{NFC\_CE2}}$ 11 GPIO20
J1850_TX	0x80	STD_PU_ST	0x00 7'b00_000_00	00 J1850_TX 01 Reserved 10 $\overline{\text{NFC\_CE3}}$ 11 I2C1_SCL
J1850_RX	0x81	STD_PU_ST	0x18 7'b00_110_00	00 J1850_RX 01 Reserved 10 NFC_$\overline{\text{RB}}$3 11 I2C1_SDA
TCK[8]		STD_PU_ST	0x04 7'b00_001_00	
TDI[8]		STD_PU	0x18 7'b00_110_00	
TDO[8]		STD_PU	0x03 7'b00_000_11	
TMS[8]		STD_PU	0x18 7'b00_110_00	
$\overline{\text{TRST}}$[8]		STD_PU	0x18 7'b00_110_00	
$\overline{\text{PORESET}}$[8]		STD_PU_ST	0x1C 7'b00_111_00	
$\overline{\text{HRESET}}$[8]		STD_PU_ST	0x1F 7'b00_111_11	
$\overline{\text{SRESET}}$[8]		STD_PU_ST	0x1F 7'b00_111_11	

[1] Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

[2] If RST_CONF_LOC[1:0] = 01, the reset value[6:0] = 7’b00_000_00.

[3] If RST_CONF_LOC[1:0] = 01, the reset value[6:0] = 7’b00_110_00.

[4] If RST_CONF_LOC[1:0] = 01, the reset value[6:0] = 7’b00_110_00. If RST_CONF_LOC[1] = 1, the reset value[6:0] = 7’b00_110_11.

[5] If RST_CONF_LOC[1:0] = 10, the reset value[6:0] = 7’b01_000_11. If RST_CONF_LOC[1:0] = 00, the reset value[6:0] = 7’b00_000_11.

[6] If RST_CONF_LOC[1:0] = 00 and RST_CONF_MX = 0, the reset value[6:0] = 7’b11_000_11.

[7] If RST_CONF_LOC[1:0] = 11, the reset value[6:0] = 7’b11_000_11.

[8] Not controlled by I/O Control registers.

Note: RST_CONF_LOC[1:0] is defined as:
2’b00: Boot from LPC.
2’b01: boot From NFC.

Note: If RST_CONF_LOC[1] = 1, the slew rate of pad is fastest, and thus the reset value[1:0] = 11.

20.3 Application Information

For GPIO, GPTimer, USB2OTG ULPI, CAN CLOCK IN, PSC, PSC_MCLK, NFC_RB, I^2C, and MSCAN, more than one pad can be used. Avoid selecting more than one pad for the same ALT*n* function (using FUNCMUX[1:0]) at the same time.

For example, I2C1_SCL can be configured to be at pad I2C1_SCL function ALT0 (FUNCMUX[1:0] = 00), or pad J1850_TX ALT3 (FUNCMUX[1:0] = 11). Never configure pad I2C1_SCL for ALT0 and J1850_TX for ALT3 at the same time.

NOTE

When booting from the NFC, the NFC_RB pin needs an external pullup resistor.

Chapter 21
LocalPlus Bus Controller (LPC)

21.1 Introduction

The LocalPlus Bus (LPC) is the external bus interface of the MPC5125. This multi-function bus system supports interfacing to external boot ROM or flash memories, external SRAM memories, or other memory mapped devices. See Figure 21-1 for a block diagram of the LPC.

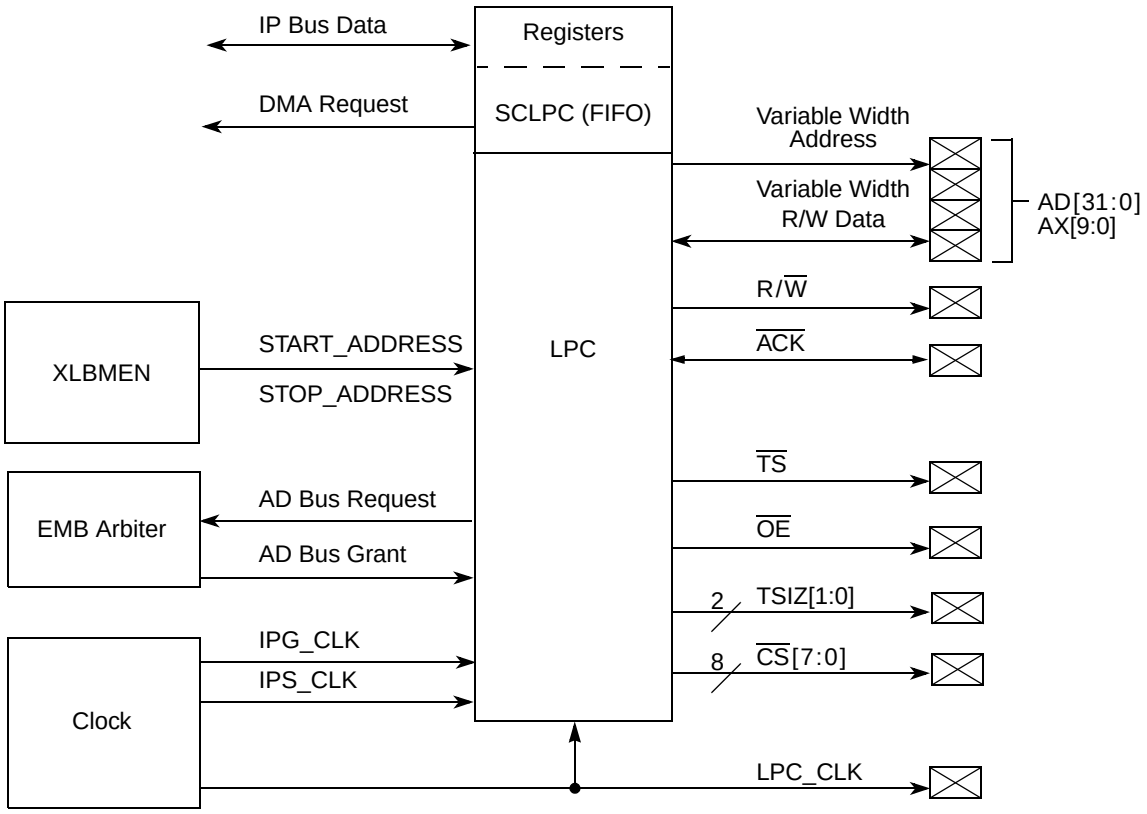


Figure 21-1. LPC Block Diagram

NOTE

AD is the shared address/data bus. AX is the address extension bus.

21.1.1 Features

The LPC includes:

- Interface to memory mapped or chip selected devices
- Two main modes of operation:
  - — Non-muxed modes
    - – Address up to 32 bits, data 8, 16, or 32 bits
  - — Muxed modes (address latch enable (ALE) used)
    - – Address up to 32 bits, data 8, 16, or 32 bits
    - – Programmable ALE level
- 8 chip-select (CS) signals
  - — Programmable wait states per CS
  - — Programmable deadcycles per CS
  - — Programmable holdcycles per CS
  - — Programmable byte swapping per CS
- Configurable boot interface supporting Power Architecture architecture code execution
- Dynamic bus sizing
- Support of burst mode flash devices (up to 32 byte bursts)
  - — Synchronous burst read
  - — Synchronous burst write
  - — Asynchronous burst read (page mode)
  - — Asynchronous burst write (page mode)
- DMA support allows data movement independently from the CPU (up to 56 byte bursts)
- No support of misaligned accesses. This also includes transfers of 3, 5, 6, and 7 bytes.

Table 21-1. Signal Properties

Name	Function	I/O	Reset	Pullup
AD[31:0]	Address and data lines	I/O	0	—
AX[31:0]	Non-muxed mode: address lines Muxed mode: address latch enable (AX[0]) Transfer size (AX[2:1]) Transfer start (AX[3])	O	0	—
$\overline{\text{ACK}}$	No burst transaction: Acknowledge can shorten a transaction Burst transaction: Indicates that a burst transaction is ongoing.	I/O	—	Pullup
LPC_CLK	LPC clock	O	—	—
$\overline{\text{CS}}$[7:0]	Chip select	O	1	—
$\overline{\text{OE}}$	Output enable	O	1	—
R/$\overline{\text{W}}$	Read/write bar	O	1	—
TSIZ[1:0]	Transfer size	O	0	—
$\overline{\text{TS}}$	Transfer start	O	0	—

21.2 Memory Map and Register Definition

Table 21-2. LPC Block Memory Map

Offset from LPC_BASE (0xFF41_0100)[1]	Register	Access[2]	Reset Value[3]	Section/Page
General Registers				
0x000	Chip Select 0/Boot Configuration Register (LPC_CS0BOOTC)	R/W	0x0020_UUU1	21.2.1.1.1/21-522
0x004	Chip Select 1 Configuration Register (LPC_CS1C)	R/W	0x0000_0000	21.2.1.1.2/21-525
0x008	Chip Select 2 Configuration Register (LPC_CS2C)	R/W	0x0000_0000	21.2.1.1.2/21-525
0x00C	Chip Select 3 Configuration Register (LPC_CS3C)	R/W	0x0000_0000	21.2.1.1.2/21-525
0x010	Chip Select 4 Configuration Register (LPC_CS4C)	R/W	0x0000_0000	21.2.1.1.2/21-525
0x014	Chip Select 5 Configuration Register (LPC_CS5C)	R/W	0x0000_0000	21.2.1.1.2/21-525
0x018	Chip Select 6 Configuration Register (LPC_CS6C)	R/W	0x0000_0000	21.2.1.1.2/21-525
0x01C	Chip Select 7 Configuration Register (LPC_CS7C)	R/W	0x0000_0000	21.2.1.1.2/21-525
0x020	Chip Select Control Register (LPC_CSC)	R/W	0x0000_0000	21.2.1.1.3/21-527
0x024	Chip Select Status Register (LPC_CSS)	R/W	0x0000_0000	21.2.1.1.4/21-527
0x028	Chip Select Burst Control (LPC_CSBC)	R/W	0x0000_0000	21.2.1.1.5/21-528
0x02C	Chip Select Deadcycle Control Register (LPC_CSDC)	R/W	0x3333_3333	21.2.1.1.6/21-529
0x030	Chip Select Holdcycle Control Register (LPC_CSHC)	R/W	0x3333_3333	21.2.1.1.7/21-530
0x034	Address Latch Timing Register (LPC_ALTR)	R/W	0x0000_0000	21.2.1.1.8/21-530
0x038	TSIZ and TS Enable Register (LPC_TTE)	R/W	0x0000_000U	21.2.1.1.9/21-531
0x03C–0x0FF	Reserved			
0x100	SCLPC Packet Size Register (LPC_SCLPC_PS)	R/W	0x0000_0000	21.2.1.2.1/21-532
0x104	SCLPC Start Address Register (LPC_SCLPC_SA)	R/W	0x0000_0000	21.2.1.2.2/21-533
0x108	SCLPC Control Register (LPC_SCLPC_C)	R/W	0x0001_0000	21.2.1.2.3/21-533
0x10C	SCLPC Enable Register (LPC_SCLPC_E)	R/W	0x0000_0000	21.2.1.2.4/21-534
0x110	Reserved			
0x114	SCLPC Status Register (LPC_SCLPC_S)	R/W	0x0000_0000	21.2.1.2.5/21-535
0x118	SCLPC Bytes Done Register (LPC_SCLPC_BD)	R/W	0x0000_0000	21.2.1.2.6/21-535
0x11C	EMB Share Counter Register (LPC_EMB_SC)	R/W	0x0010_0000	21.2.1.2.7/21-536
0x120	EMB Pause Control Register (LPC_EMB_PC)	R/W	0x0000_0000	21.2.1.2.8/21-536
0x124–0x13F	Reserved			
0x140	LPC RX/TX FIFO Data Word Register (LPC_SCLPC_FIFOD)	R/W	—[4]	21.2.1.3.1/21-537
0x144	LPC RX/TX FIFO Status Register (LPC_SCLPC_FIFOS)	R/W	0x0001_0000	21.2.1.3.2/21-538
0x148	LPC RX/TX FIFO Control Register (LPC_SCLPC_FIFOC)	R/W	0x0100_0000	21.2.1.3.3/21-539
0x14C	LPC RX/TX FIFO Alarm Register (LPC_SCLPC_FIFOA)	R/W	0x0000_0000	21.2.1.3.4/21-539
0x150–0x1FF	Reserved			

[1] Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

[2] In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

[3] In this column, the symbol “U” indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

[4] Reset value is indeterminate.

21.2.1 Register Descriptions

21.2.1.1 Chip Select/LPC Registers—0x0000

The following registers are available:

- Section 21.2.1.1.1, “Chip Select 0/Boot Configuration (LPC_CS0BOOTC) Register” (0x000)
- Section 21.2.1.1.2, “Chip Select[1:7] Configuration (LPC_CSnC) Registers” (0x004–0x001C)
- Section 21.2.1.1.3, “Chip Select Control (LPC_CSC) Register” (0x020)
- Section 21.2.1.1.4, “Chip Select Status (LPC_CSS) Register” (0x024)
- Section 21.2.1.1.5, “Chip Select Burst Control (LPC_CSBC) Register” (0x008)
- Section 21.2.1.1.6, “Chip Select Deadcycle Control (LPC_CSDC) Register” (0x02C)
- Section 21.2.1.1.7, “Chip Select Holdcycle Control (LPC_CSHC) Register” (0x030)
- Section 21.2.1.1.8, “Address Latch Timing (LPC_ALTR) Register” (0x034)
- Section 21.2.1.1.9, “TSIZ and TS Enable (LPC_TTE) Register” (0x038)

21.2.1.1.1 Chip Select 0/Boot Configuration (LPC_CS0BOOTC) Register

Address: Base + 0x000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WaitP								WaitX							
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MX	ALEV	AA	CE	ALEN		DS		BM	ADD RM	WTyp		WS	RS	WO	RO
Reset	—[1,2]	0	0	1	0	0	—[1,2]	—[1,2]	0	—[1,2]	0	0	0	0	0	1

Figure 21-2. Chip Select 0/Boot Configuration (LPC_CS0BOOTC) Register

[1] The reset value is defined by the latched reset configuration word (RST_CONF). See Table 4-2.

[2] See bitfield description in Table 21-3.

Table 21-3. LPC_CS0BOOTC field descriptions

Field	Description
WaitP	Number of wait states to insert. Can be applied as a prescale to WaitX or used by itself, as dictated by the WTyp bits (see below). Wait states control the number of LPC clocks for which the corresponding CS pin remains active in a non-burstable transaction. The default wait time is two LPC clocks. For example, if the WaitP is set to four, the CS is asserted as maximum for six clocks. Acknowledge can shorten the WaitP time, but not the fixed two LPC clock time. This parameter describes the time before the burst signal is asserted for a burst transaction. An additional two clocks are also available for this read operation. There are an additional two and one-half clocks available for write operations.
WaitX	The base number of wait states to insert, or to be combined with WaitP, as dictated by the WTyp bits (see below). See the WaitP description.
MX	The MX bit specifies whether a transaction operates a muxed or non-muxed. A muxed transaction presents address and data in different tenures. ALE is asserted during the address tenure. At the end of ALE, the address remains driven for at least one LPC clock before the $\overline{CSx}$ pin is asserted. 0 Non-muxed. 1 Muxed. **Note:** The reset value of MX is RST_CONF_LPCMX. See Section 4.5, “Reset Configuration Word (RST_CONF),” for more information.
ALEV	ALE level. 0 ALE is active low. 1 ALE is active high.
AA	ACK active. This bit defines whether $\overline{ACK}$ input is active or not. 0 $\overline{ACK}$ input is not active and cannot shorten the wait state time. 1 Programmed wait states can be overridden if/when the external device drives the $\overline{ACK}$ input low. Wait states remain in effect. If no $\overline{ACK}$ is received, the cycle terminates at the end of the wait state period.
CE	An individual enable bit that allows CS operation for the corresponding CS pin. CE must be high to allow operation. The chip select control register ME bit must also be high, except when CS[0] is used for boot ROM. 1 External CS is enabled. 0 External CS is disabled.
ALEN	ALE length 00 ALE width is 1 LPC clock. 01 ALE width is 2 LPC clocks. 10 ALE width is 3 LPC clocks. 11 ALE width is 4 LPC clocks. **Note:** ALE length configures not only the width of the ALE assertion, but also the width of the isolation cycle between ALE deassertion and CS assertion.
DS	Data size field, which represents the device data bus size (in bytes): 00 1byte. 01 2 bytes. 10 Reserved. 11 4bytes. **Note:** Table 21-26 and Table 21-30 show on which AD lines the data is located. **Note:** The reset value of DS is RST_CONF_LPC_DBW. See Section 4.5, “Reset Configuration Word (RST_CONF),” for more information.

Table 21-3. LPC_CS0BOOTC field descriptions (Continued)

Field	Description
BM	Burst mode 0 Synchronous burst mode. 1 Asynchronous burst mode (page mode). **Note:** The asynchronous burst mode setting is only valid for non-muxed transactions. If this bit is set and muxed mode is enabled, a synchronous burst is performed. **Note:** If bursting (read or write) is not enabled, this bit has no effect.
ADDRM	Address mode 0 Byte addressing. 1 Short or word addressing. An 8-bit data bus always uses byte addressing. **Note:** Table 21-26 and Table 21-30 show on which AD/AX lines the address is located. **Note:** The reset value of ADDRM is RST_CONF_LPCWA. See Section 4.5, "Reset Configuration Word (RST_CONF)," for more information.
WTyp	Wait state type bits that define the application of wait states contained in WaitP and WaitX fields as follows: 00 WaitX is applied to read and write cycles (WaitP is ignored). 01 WaitX is applied to read cycles; WaitP is applied to write cycles. 10 WaitX is applied to reads; WaitP/WaitX (16-bit value) is applied to writes. 11 WaitP/WaitX (as a full 16-bit value) is applied to reads and writes.
WS	Write swap bit. If high, endian byte swapping occurs during writes to a device. • For 8-bit devices, this bit has no effect. • For 16-bit devices, byte swapping can occur. • For 32-bit devices (possible in muxed mode only), byte swapping can occur. 0 Swapping cannot occur. 1 Swapping can occur. A 2-byte swap is AB to BA; a 4-byte swap is ABCD to DCBA. **Note:** Transactions at less than the defined port size (i.e., data size) apply swapping rules as above, according to the current transaction size.
RS	Read swap bit. Same as WS, but swapping is done when reading data from a device. 0 Swapping cannot occur. 1 Swapping can occur. **Note:** Transactions at less than the defined port size (i.e., data size) apply swapping rules as above, according to the current transaction size.
WO	Write-only bit. If the bit is high, the device is treated as a write-only device. An attempted read access can results in an interrupt (as dictated by the Chip Select Control (LPC_CSC) Register IE bit). In any case, no transaction is presented to the device.
RO	Read-only bit. If the bit is high, the device is treated as a read-only device. An attempted write access can result in an interrupt (as dictated by the Chip Select Control (LPC_CSC) Register IE bit). In any case, no transaction is presented to the device. **Note:** This bit is high from reset, indicating the boot device is read-only.

21.2.1.1.2 Chip Select[1:7] Configuration (LPC_CS*n*C) Registers

Address: Base + 0x004 (Chip Select Configuration Register 1)
Base + 0x008 (Chip Select Configuration Register 2)
Base + 0x00C (Chip Select Configuration Register 3)
Base + 0x010 (Chip Select Configuration Register 4)
Base + 0x014 (Chip Select Configuration Register 5)
Base + 0x018 (Chip Select Configuration Register 6)
Base + 0x01C (Chip Select Configuration Register 7)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WaitP								WaitX							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MX	ALEV	AA	CE	ALEN		DS		BM	ADD RM	WTyp		WS	RS	WO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-3. Chip Select[1:7] Configuration (LPC_CS*n*C) Registers

Table 21-4. LPC_CS*n*C field descriptions

Field	Description
WaitP	Number of wait states to insert. Can be applied as a prescale to WaitX or used by itself, as dictated by the WTyp bits (see below). Wait states control the number of LPC clocks for which the corresponding CS pin remains active in a non-burstable transaction. The default wait time is two LPC clocks. For example, if the WaitP is set to four, the CS is asserted as maximum for six clocks. Acknowledgment can shorten the WaitP time, but not the fixed two LPC clock time. This parameter describes the time before the burst signal is asserted for a burst transaction. An additional two clocks are also available for this read operation. There are an additional two and one-half clocks available for write operations.
WaitX	The base number of wait states to insert, or to be combined with WaitP, as dictated by the WTyp bits (see below). See the WaitP description.
MX	The MX bit specifies whether a transaction operates as muxed or non-muxed. A muxed transaction presents address and data in different tenures. ALE is asserted during the address tenure. At the end of ALE, the address remains driven for at least one LPC clock before the $\overline{\text{CSx}}$ pin is asserted. 0 Non-muxed. 1 Muxed.
ALEV	ALE level 0 ALE is active low. 1 ALE is active high.
AA	ACK active. This bit defines whether $\overline{\text{ACK}}$ input is active or not. 0 $\overline{\text{ACK}}$ input is not active and cannot shorten the wait state time. 1 Programmed wait states can be overridden if/when the external device drives the $\overline{\text{ACK}}$ input low. Wait states remains in effect. If no $\overline{\text{ACK}}$ is received, the cycle terminates at the end of wait state period.
CE	An individual enable bit that allows CS operation for the corresponding CS pin. CE must be high to allow operation. The chip select control register ME bit must also be high, except when CS[0] is used for boot ROM. 0 External CS is disabled. 1 External CS is enabled.

Table 21-4. LPC_CS*n*C field descriptions (Continued)

Field	Description
ALEN	ALE length 00 ALE width is one LPC clock. 01 ALE width is two LPC clocks. 10 ALE width is three LPC clocks. 11 ALE width is four LPC clocks. **Note:** ALE length configures not only the width of the ALE assertion, but also the width of the isolation cycle between ALE deassertion and CS assertion.
DS	Data size field that represents the device data bus size (in bytes): 00 1byte. 01 2 bytes. 10 Reserved. 11 4 bytes. **Note:** Table 21-26 and Table 21-30 show on which AD lines the data is located.
BM	Burst mode 0 Synchronous burst mode. 1 Asynchronous burst mode (page mode). **Note:** The asynchronous burst mode setting is only valid for non-muxed transactions. If this bit is set and muxed mode is enabled, a synchronous burst is performed. **Note:** This bit has no influence if bursting (read or write) is not enabled.
ADDRM	Address mode 0 Byte addressing. 1 Short or word addressing. An 8-bit data bus always uses a byte addressing. **Note:** Table 21-26 and Table 21-30 show on which AD/AX lines the address is located.
WTyp	Wait state type bits that define the application of wait states contained in the WaitP and WaitX fields, as follows: 00 WaitX is applied to read and write cycles (WaitP is ignored) 01 WaitX is applied to read cycles; WaitP is applied to write cycles 10 WaitX is applied to reads; WaitP/WaitX (16-bit value) is applied to writes 11 WaitP/WaitX (as a full 16-bit value) is applied to reads and writes
WS	Write swap bit. If high, endian byte swapping occurs during writes to a device. • For 8-bit devices, this bit has no effect • For 16-bit devices, byte swapping can occur • For 32-bit devices (possible in muxed mode only), byte swapping can occur 0 Swapping cannot occur. 1 Swapping can occur. A 2-byte swap is AB to BA; a 4-byte swap is ABCD to DCBA. **Note:** Transactions at less than the defined port size (i.e., data size) apply swapping rules as above, according to the current transaction size.
RS	Read swap bit. Same as WS, but swapping is done when reading data from a device. 0 Swapping cannot occur. 1 Swapping can occur. **Note:** Transactions at less than the defined port size (i.e., data size) apply swapping rules as above, according to the current transaction size.
WO	Write-only bit. If the bit is high, the device is treated as a write-only device. An attempted read access can results in an interrupt (as dictated by the Chip Select Control (LPC_CSC) Register IE bit). In any case, no transaction is presented to the device.

Table 21-4. LPC_CS*n*C field descriptions (Continued)

Field	Description
RO	Read only bit. If the bit is high, the device is treated as a read-only device. An attempted write access can results in an interrupt (as dictated by the Chip Select Control (LPC_CSC) Register IE bit). In any case, no transaction is presented to the device.

21.2.1.1.3 Chip Select Control (LPC_CSC) Register

Address: Base + 0x020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	IE	0	0	ME	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-4. Chip Select Control (LPC_CSC) Register

Table 21-5. CSC field descriptions

Field	Description
IE	Interrupt enable bit. An interrupt can be generated if a write is initiated to a read-only define CS; a read is initiated to a write-only defined CS; an SCLPC state machine finishes transfer (normal or abort); or an SCLPC FIFO detects FIFO errors (underrun or overrun).
ME	Master enable bit that is a global module enable bit. If this bit is low, register access can continue to occur, but no external transactions are accepted. However, ME does not affect boot ROM operation on $\overline{CS}$[0]. For software to disable $\overline{CS}$[0], it must write 0 to the chip select boot ROM configuration register enable bit (CE).

21.2.1.1.4 Chip Select Status (LPC_CSS) Register

Address: Base + 0x024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	WOerr	ROerr	0	CSxerr			0	0	0	0	0	0	0	0
W			w1c	w1c												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-5. Chip Select Status (LPC_CSS) Register

Table 21-6. LPC_CSS field descriptions

Field	Description
WOerr	Write-Only Error. If 1, it indicates a read access was attempted on a device marked as write-only. This is a sticky bit and must be written with 1 to be cleared. This status bit is always active, regardless of the bus error enable bit. The CS number that relates to the error is reflected in the CSxerr field. An interrupt is also generated if the IE bit is set.
ROerr	Read-Only Error. If 1, it indicates a write access was attempted on a device marked as read-only. This is a sticky bit and must be written with 1 to be cleared. This status bit is always active, regardless of the bus error enable bit. The CS number that relates to the error is reflected in the $\overline{CS}$xerr field. An interrupt is also generated if the IE bit is set.
CSxerr	Chip select error that indicates CS number associated with last Write-Only or Read-Only Error, as long as only one error type happens (write- or read-only). When the other type is happening, CSxerr points to the first violating CS of the opposite type. This is the case until one of the Error flags gets cleared.

21.2.1.1.5 Chip Select Burst Control (LPC_CSBC) Register

Address: Base + 0x028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CW7	SLB7	BWE7	BRE7	CW6	SLB6	BWE6	BRE6	CW5	SLB5	BWE5	BRE5	CW4	SLB4	BWE4	BRE4
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CW3	SLB3	BWE3	BRE3	CW2	SLB2	BWE2	BRE2	CW1	SLB1	BWE1	BRE1	CW0	SLB0	BWE0	BRE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-6. Chip Select Burst Control (LPC_CSBC) Register

Table 21-7. LPC_CSBC field descriptions

Field	Description
CW7	Chip select 7 cache wrap-capable; set if a device burst can perform PPC cache wrap. **Note:** Cache wrap means the external device supports a wrap-around mechanism at addresses 0x20, 0x40, 0x60,... and so on. For example, 0x8-0xC-0x10-0x14-0x18-0x1C-0x0-0x4.
SLB7	Chip select 7 short/long burst; set 0 for short burst only, 1 for long burst-capable. Short burst are limited to 8 bytes, used for instruction fetches. Long burst-capable means the device can do a 16-, 24-, 32-, 40-, 48-, or 56-byte burst. The length of the burst depends on the amount of data which should be transferred at once.
BWE7	Chip select 7 burst write enable; set 1 to enable device bursting for a given chip select. This bit must be set to enable any bursting writes.
BRE7	Chip select 7 burst read enable, 1 to enable device bursting for given chip select. Must be set to enable bursting reads.

Table 21-7. LPC_CSBC field descriptions (Continued)

Field	Description
CW[6:0]	Same as CW7, but for CS6–CS0/Boot.
SLB[6:0]	Same as SLB7, but for CS6–CS0/Boot.
BWE[6:0]	Same as BWE7, but for CS6–CS0/Boot.
BRE[6:0]	Same as BRE7, but for CS6–CS0/Boot.

21.2.1.1.6 Chip Select Deadcycle Control (LPC_CSDC) Register

Address: Base + 0x02C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	DC7		0	0	DC6		0	0	DC5		0	0	DC4	
W																
Reset	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	DC3		0	0	DC2		0	0	DC1		0	0	DC0	
W																
Reset	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

Figure 21-7. Chip Select Deadcycle Control (LPC_CSDC) Register

Table 21-8. LPC_CSDC field descriptions

Field	Description
DC7	Deadcycles can be specified as 0 to 3. Deadcycles are added to the end of a Chip Select 7 read access and occur in addition to any cycles that may already exist. These cycles provide the device additional time to tri-state its bus after a read operation. 00 Device can drive data one LPC clock cycle after CS deassertion. 01 Device can drive data two LPC clock cycle after CS deassertion. 10 Device can drive data three LPC clock cycle after CS deassertion. 11 Device can drive data four LPC clock cycle after CS deassertion.
DC[6:0]	Same as DC7, but for CS6–CS0/Boot.

21.2.1.1.7 Chip Select Holdcycle Control (LPC_CSHC) Register

Address: Base + 0x030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	HC7		0	0	HC6		0	0	HC5		0	0	HC4	
W																
Reset	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	HC3		0	0	HC2		0	0	HC1		0	0	HC0	
W																
Reset	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

Figure 21-8. Chip Select Holdcycle Control (LPC_CSHC) Register

Table 21-9. LPC_CSHC field descriptions

Field	Description
HC7	Holdcycles can be specified as 0 to 3. Holdcycles are added to the end of a Chip Select 7 write access and occur in addition to any cycles that may already exist. These cycles provide the device additional time to latch the data from the bus after a write operation. 00 Data is valid one LPC clock cycle after CS deassertion. 01 Data is valid two LPC clock cycle after CS deassertion. 10 Data is valid three LPC clock cycle after CS deassertion. 11 Data is valid four LPC clock cycle after CS deassertion. **Note:** For a write burst transaction the data is valid a half cycle less.
HC[6:0]	Same as HC7, but for CS6–CS0/Boot.

21.2.1.1.8 Address Latch Timing (LPC_ALTR) Register

Address: Base + 0x034 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	ALT7	ALT6	ALT5	ALT4	ALT3	ALT2	ALT1	ALT0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-9. Address Latch Timing (LPC_ALTR) Register

Table 21-10. LPC_ALTR field descriptions

Field	Description
ALT7	Chip Select 7 address latch timing modification for multiplexed mode. 0 CS is asserted together with the assertion of Address latch (ALE). 1 CS is asserted (ALEN + 1) x LPC_CLK clocks after the deassertion of ALE.
ALT[6:0]	Same as ALT7, but for CS6–CS0/Boot.

21.2.1.1.9 TSIZ and TS Enable (LPC_TTE) Register

Address: Base + 0x038 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TSE	TSIZE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	—[1,2]	—[1,2]

Figure 21-10. TSIZ and TS Enable (LPC_TTE) Register

1 The reset value is defined by the latched reset configuration word (RST_CONF). See Table 4-2.

2 See bit description in Table 21-11.

Table 21-11. LPC_TTE field descriptions

Field	Description
TSE	Transfer Start Enable bit. The reset value is RST_CONF_LPC_TS. See Section 4.5, “Reset Configuration Word (RST_CONF).” 0 Transfer Start Signal is driven high. 1 Transfer Start Signal is brought out.
TSIZE	Transfer Size Enable bit. The reset value is RST_CONF_LPC_TS. See Section 4.5, “Reset Configuration Word (RST_CONF).” 0 Transfer Size Signal is driven high. 1 Transfer Size Signal is brought out.

21.2.1.2 SCLPC Registers—0x0100

There are seven 32-bit registers for the LPC (SCLPC).

The following registers are available:

- Section 21.2.1.2.1, “SCLPC Packet Size (LPC_SCLPC_PS) Register” (0x0100)
- Section 21.2.1.2.3, “SCLPC Control (LPC_SCLPC_C) Register” (0x0108)

- Section 21.2.1.2.4, “SCLPC Enable (LPC_SCLPC_E) Register” (0x010C)
- Section 21.2.1.2.5, “SCLPC Status (LPC_SCLPC_S) Register” (0x0114)
- Section 21.2.1.2.6, “SCLPC Bytes Done (LPC_SCLPC_BD) Register” (0x0118)
- Section 21.2.1.2.7, “EMB Share and Wait Count (LPC_EMB_SC) Register” (0x011C)
- Section 21.2.1.2.8, “EMB Pause Control (LPC_EMB_PC) Register” (0x0120)

21.2.1.2.1 SCLPC Packet Size (LPC_SCLPC_PS) Register

Address: Base + 0x100 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PACKET SIZE														
W	RESTART															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PACKET SIZE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-11. SCLPC Packet Size (LPC_SCLPC_PS) Register

Table 21-12. LPC_SCLPC_PS field descriptions

Field	Description
RESTART	Writing a 1 to this bit begins a SCLPC transfer. It clears automatically and always reads back as 0. **Note:** Start transfers after LPC FIFO and SCLPC are configured.
PACKET SIZE	This 31-bit field represents the number of bytes SCLPC must transact before going idle and waiting for a restart. **Note:** The co-location of the restart bit and the packet_size field allows software to restart a transaction and change the packet_size in a single write. Maximum packet size is 2 GB – 1 bytes. **Note:** PACKET SIZE needs to be either a multiple of BPT setting (see Table 21-15) or of 8.

21.2.1.2.2 SCLPC Start Address (LPC_SCLPC_SA) Register

Address: Base + 0x104 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	START ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	START ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-12. SCLPC Start Address (LPC_SCLPC_SA) Register

Table 21-13. LPC_SCLPC_SA field descriptions

Field	Description
START ADDRESS	The address of the first byte in the packet to be sent. **Note:** START ADDRESS needs to be either a multiple of BPT setting (see Table 21-15) or of 8.

21.2.1.2.3 SCLPC Control (LPC_SCLPC_C) Register

Address: Base + 0x108 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	CSX			0	0	0	0	0	0	FLUSH	R/$\overline{W}$
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	DAI	0	0	BPT					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-13. SCLPC Control (LPC_SCLPC_C) Register

Table 21-14. LPC_SCLPC_C field descriptions

Field	Description
CSX	This field should be written with the chip select number associated with each SCLPC transaction. **Note:** LPC configuration registers associated with this CS also affect SCLPC transactions. The two work together.
FLUSH	If set to 1, this bit enables the assertion of the DMA request at the completion of a read packet, regardless of the actual state of the physical FIFO alarm. The DMA request deasserts after the FIFO empties.
R/$\overline{W}$	Read/Write bar that controls the direction of the SCLPC transaction. 0 SCLPC writes to the device; i.e., FIFO transmit. 1 SCLPC reads from the device; i.e., FIFO receive. (Default)

Table 21-14. LPC_SCLPC_C field descriptions (Continued)

Field	Description
DAI	Disable auto increment. Normally, SCLPC and LPC present sequential incrementing addresses to the device as the packet proceeds. If the device is operating as a single address FIFO, the DAI bit should be set to 1. When set, addresses to the device are stuck at start_address for every transaction. For DAI operation, the BPT field must be set to the port size of the device.
BPT	Bytes per transaction that indicates the number of bytes per transaction. The only valid entries in this field are 1, 2, 4, 8, 16, 24, 32, 40, 48, and 56. Start_address and packet_size values must be aligned/multiples of BPT or multiples of 8. BPT should be set to the device port size for DAI operation.

21.2.1.2.4 SCLPC Enable (LPC_SCLPC_E) Register

Address: Base + 0x10C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	RC	0	0	0	0	0	0	0	RF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	AIE	NIE	0	0	0	0	0	0	0	ME
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-14. SCLPC Enable (LPC_SCLPC_E) Register

Table 21-15. LPC_SCLPC_E field descriptions

Field	Description
RC	Reset controller. This bit allows for a software reset of the SCLPC state machine. Writing a 1 to this bit resets the SCLPC state machine. Reset is maintained as long as this bit is high. Software must write this bit low to release the reset and start operation. **Note:** Although RC does not reset this register interface, it does clear interrupt and interrupt status conditions. Never reset the SCLPC Controller during a transaction (TX or RX).
RF	Reset FIFO. This is the FIFO software reset bit. Writing a 1 to this bit resets the SCLPC FIFO. For normal operation, the FIFO must not be in reset. Resetting the FIFO clears the FIFO of data and resets its read/write pointers and the status bits, but it does not disturb previously programmed alarm and granularity settings. **Note:** It is recommended that software set and clear the RC and RF bits prior to programming and starting a packet.
AIE	Abort interrupt enable. If set, and a FIFO error occurs during packet transmission, a CPU interrupt from SCLPC is generated. In any case, the packet is terminated and an abort status bit is set.
NIE	Normal interrupt enable. If set, this bit enables a CPU interrupt to occur at the end of a normally terminated packet. There is also an NT status bit that sets in any case.
ME	Master enable. This bit must be set to 1 to generate a restart to the SCLPC state machine. Restart is achieved by writing 1 to byte 0 of the packet_size register. This ME bit must also be set for a restart to occur. **Note:** If ME is low (inactive), it also clears interrupt and interrupt status.

21.2.1.2.5 SCLPC Status (LPC_SCLPC_S) Register

Address: Base + 0x114 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	AT	0	0	0	NT	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-15. SCLPC Status (LPC_SCLPC_S) Register

Table 21-16. LPC_SCLPC_S field descriptions

Field	Description
AT	Abort termination. This bit is set to 1 if the packet has terminated abnormally (which is only possible if a FIFO error occurred). **Note:** This bit is ANDed with the AIE bit (see Table 21-15) to generate a single CPU interrupt signal to the core. This bit is sticky. Write to 1 for clearing the bit and clearing the interrupt. **Note:** This bit (and any interrupt) is also cleared if: 1) the RC bit is set, 2) the ME bit is clear, or 3) restart occurs.
NT	Normal termination. This bit is set to 1 when a complete packet has been transferred successfully. **Note:** This bit is ANDed with the NIE bit (see Table 21-15) to generate a single CPU interrupt signal to the core. This bit is sticky. Write to 1 for clearing the bit and clearing the interrupt.

21.2.1.2.6 SCLPC Bytes Done (LPC_SCLPC_BD) Register

Address: Base + 0x118 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BYTES DONE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BYTES DONE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-16. SCLPC Bytes Done (LPC_SCLPC_BD) Register

Table 21-17. LPC_SCLPC_BD field descriptions

Field	Description
BYTES DONE	BYTES DONE is updated dynamically by the SCLPC state machine to represent the actual number of bytes transmitted at a given point in time. At the normal conclusion of a packet, the BYTES DONE field should match the PACKET_SIZE field.

21.2.1.2.7 EMB Share and Wait Count (LPC_EMB_SC) Register

The EMB Share and Wait Count (LPC_EMB_SC) register configures the EMB arbiter and belongs to the LPC functionality.

Address: Base + 0x11C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EMB_SHARE_COUNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	0	NFC_WAIT_COUNT					0					LPC_WAIT_COUNT				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-17. EMB Share and Wait Count (LPC_EMB_SC) Register

Table 21-18. LPC_EMB_SC field descriptions

Field	Description
EMB_SHARE_COUNT	This 16-bit value controls the length of the time slot assigned to NFC transactions before an SCLPC or CSB LPC (if the LPC_P bit is set in the EMB pause control register) request starts to pause the other modules.
NFC_WAIT_COUNT	This 5-bit value controls how long the bus remains assigned to NFC after the bus goes idle.
LPC_WAIT_COUNT	This 5-bit value controls how long the bus remains assigned to LPC after the bus goes idle.

21.2.1.2.8 EMB Pause Control (LPC_EMB_PC) Register

The EMB Pause Control (LPC_EMB_PC) register configures the EMB arbiter and belong to the LPC functionality.

Address: Base + 0120 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LPC_P
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-18. EMB Pause Control (LPC_EMB_PC) Register

Table 21-19. LPC_EMB_PC field descriptions

Field	Description
LPC_P	LPC CSB pause disable 0 LPC CSB request immediately pauses an NFC transfer. 1 LPC CSB request does not immediately pause an NFC transfer. The pause is initiated after the EMB_SHARE_COUNT expires or noNFC request is asserted.

21.2.1.3 LPC RX/TX FIFO Registers

LPC uses a single FIFO that changes direction based on the RX/TX mode. Software controls direction change and flushes FIFO before changing directions. FIFO memory is 1024 bytes (256 x 32).

LPC FIFO is controlled by four 32-bit registers. The following LPC FIFO registers are provided:

- Section 21.2.1.3.1, “LPC RX/TX FIFO Data Word (LPC_SCLPC_FIFOD) Register” (0x0140)
- Section 21.2.1.3.2, “LPC RX/TX FIFO Status (LPC_SCLPC_FIFOS) Register” (0x0144)
- Section 21.2.1.3.3, “LPC RX/TX FIFO Control (LPC_SCLPC_FIFOC) Register” (0x0148)
- Section 21.2.1.3.4, “LPC RX/TX FIFO Alarm (LPC_SCLPC_FIFOA) Register” (0x014C)

21.2.1.3.1 LPC RX/TX FIFO Data Word (LPC_SCLPC_FIFOD) Register

Address: Base + 0x140 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FIFO_Data_Word															
Reset[1]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	FIFO_Data_Word															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 21-19. LPC RX/TX FIFO Data Word (LPC_SCLPC_FIFOD) Register

[1] Reset value is indeterminate.

Table 21-20. LPC_SCLPC_FIFOD field descriptions

Field	Description
FIFO_Data_Word	The FIFO data port. Reading from this location fetches data from the FIFO writing to this location writes data into the FIFO. During normal operation, the DMA controller moves data to and from this register. **Note:** Only full-word access is allowed. If all byte enables are not asserted when accessing this location, a FIFO error flag is generated.

21.2.1.3.2 LPC RX/TX FIFO Status (LPC_SCLPC_FIFOS) Register

Address: Base + 0x144 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	ERR	UF	OF	0	FULL	ALARM	EMPTY
W										w1c	w1c	w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-20. LPC RX/TX FIFO Status (LPC_SCLPC_FIFOS) Register

Table 21-21. LPC_SCLPC_FIFOS field descriptions

Field	Description
ERR	Error. The flag bit is essentially the logical OR of UF and OF bits and can be polled to detect any FIFO error. After clearing the offending condition, writing 1 to this bit clears the flag.
UF	Underflow. The flag indicates the read pointer has surpassed the write pointer. FIFO was read beyond empty. Resetting FIFO clears this condition; writing 1 to this bit clears the flag.
OF	Overflow. The flag indicates the write pointer has surpassed the read pointer. FIFO was written beyond full. Resetting FIFO clears this condition; writing 1 to this bit clears the flag.
FULL	FIFO full. A full indication tracks with FIFO state.
ALARM	This bit is set when the FIFO level is at or below (write)/above (read) the alarm watermark, as written according to the LPC RX/TX FIFO Alarm (LPC_SCLPC_FIFOA) Register setting. This bit is cleared when the FIFO level is at or above (write)/below (read) the granularity watermark, as configured according to the LPC RX/TX FIFO Control (LPC_SCLPC_FIFOC) Register setting. Setting this bit automatically signals the DMA engine to refill (write)/empty (read) the FIFO.
EMPTY	FIFO empty. An empty indication tracks with FIFO state.

21.2.1.3.3 LPC RX/TX FIFO Control (LPC_SCLPC_FIFOC) Register

Address: Base + 0x148 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	GR			0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-21. LPC RX/TX FIFO Control (LPC_SCLPC_FIFOC) Register

Table 21-22. LPC_SCLPC_FIFOC field descriptions

Field	Description
GR	Granularity. These bits control the high (write)/low (read) watermark point at which the FIFO negates the alarm condition (i.e., a request for data). It represents the number of free bytes times four. 000 FIFO waits to become completely full (write)/empty (read) before stopping the data request. 001 FIFO stops the data request when only one word of space (write)/one word (read) remains.

21.2.1.3.4 LPC RX/TX FIFO Alarm (LPC_SCLPC_FIFOA) Register

Address: Base + 0x14C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	ALARM_W									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-22. LPC RX/TX FIFO Alarm (LPC_SCLPC_FIFOA) Register

Table 21-23. LPC_SCLPC_FIFOA field descriptions

Field	Description
ALARM_W	Alarm Watermark. Write these bits to set the low (write)/high (read) level watermark, which is the point where FIFO asserts a request for the DMA controller data to fill (write)/empty (read). The value is in bytes. For example, during a SCLPC write operation (R/$\overline{W}$ bit of SCLPC Control (LPC_SCLPC_C) Register) and an alarm watermark setting of 128, an alarm condition occurs when FIFO contains 128 bytes or less. After asserted, an alarm does not negate until a high (write)/ low (read) level mark is reached, as specified by the LPC RX/TX FIFO Control (LPC_SCLPC_FIFOC) Register granularity bits.

21.3 Functional Description

There are two primary modes of operation:

- Muxed
- Non-muxed

Within each mode, there is considerable flexibility to control the operation.

Each CS can be programmed to a different mode of operation (muxed, non-muxed, number of wait states, byte swapping, etc.).

In muxed mode, the same 32-bit local bus presents an address in an address tenure and data in a data tenure, in a muxed fashion (similar to PCI protocol).

Muxed mode provides an ALE during the address phase to capture the address. This mode requires external logic to latch the address during the address tenure. The level of the ALE can be programmed by the ALE bit.

An $\overline{ACK}$ input is provided and can be asserted to shorten (but not extend) wait states.

The LPC on MPC5125 provides an output enable signal, $\overline{OE}$, to achieve a complete glueless interface for most devices. $\overline{OE}$ is asserted one clock after the CS assertion if a read transaction occurs.

Muxed and non-muxed modes support a variety of device configurations and are configurable on a per CS basis. The read and write burst functionality is available in both modes. In non-muxed mode, an asynchronous burst (page mode burst) is also possible.

21.3.1 Non-Muxed Mode

In non-muxed mode, the 32-bit address/data bus is divided into address and/or data lines.

Table 21-24. Non-Muxed Mode Options

Data Size	Address Bus Width	Pins Used	Memory Size	Comments
8	32	40	4 GB	
16	26	42	128 MB	Short addressing
16	26	42	64 MB	Byte addressing

Table 21-24. Non-Muxed Mode Options

Data Size	Address Bus Width	Pins Used	Memory Size	Comments
32	10	42	4 KB	Word addressing
32	10	42	1 KB	Byte addressing

NOTE

The 24-bit data width is not supported.

The total pin number also requires the addition of the control signals CS, R/$\overline{W}$, $\overline{ACK}$, $\overline{OE}$, $\overline{TS}$ (programmable), and TSIZ (programmable) where available.

Table 21-25. Internal Data and Address Bus Assignment to External Signals in Non-Muxed Mode

Signal Name	8-Bit Data Bus	16-Bit Data Bus		32-Bit Data Bus	
		Byte Addressing	Short Addressing	Byte Addressing	Word Addressing
AX[09:08]	0	address[25:24]	address[26:25]	address[9:8]	address[11:10]
AX[07:00]	address[31:24]	address[23:16]	address[24:17]	address[7:0]	address[9:2]
AD[31:24]	address[23:16]	address[15:8]	address[16:9]	data[31:24]	
AD[23:16]	address[15:8]	address[7:0]	address[8:1]	data[23:16]	
AD[15:8]	address[7:0]	data[15:8]		data[15:8]	
AD[7:0]	data[7:0]	data[7:0]		data[7:0]	

TSIZ bits can be enabled for every mode, but the TSIZ bits only make sense for the byte addressing mode. The low address bits, are not available in the short or word addressing mode. Only TSIZEs of 1, 2, or 4 are supported.

TSIZ[1:0] are driven as follows:

01 = Transaction is 1 byte

10 = Transaction is 2 bytes

00 = Transaction is 4 bytes

Other values are invalid and should not be required by the external device.

Table 21-26 and Table 21-27 describes the various combinations of TSIZ. In addition also the influence of the swap bits - WS and RS of Chip Select X Configuration Register - is shown.

Table 21-26. Aligned Data Transfers for 32-Bit Data Bus Width

Swap WS or RS set	Transfer Size	TSIZ[1:0]	Address[1:0]	Data Lanes			
				AD[31:24]	AD[23:16]	AD[15:8]	AD[7:0]

Table 21-26. Aligned Data Transfers for 32-Bit Data Bus Width (Continued)

NO/YES[1]	1 byte	01	00	Data	—	—	—
			01	—	Data	—	—
			10	—	—	Data	—
			11	—	—	—	Data
NO	2 bytes	10	00	Data1	Data2	—	—
			10	—	—	Data1	Data2
YES			00	Data2	Data1	—	—
			10	—	—	Data2	Data1
NO	4 bytes	00	00	Data1	Data2	Data3	Data4
YES				Data4	Data3	Data2	Data1

Note: Data1 is most significant byte and Data2 (2 byte transfer) or Data 4 (4 byte transfer) is lowest significant byte of data word.

1 Swap setting has no influence on byte transfers.

Table 21-27. Aligned Data Transfers for 16-Bit Data Bus Width

Swap WS or RS set	Transfer Size	TSIZ0	Address 0	Data Lanes	
				AD[15:8]	AD[7:0]
NO/YES[1]	1 byte	1	0	Data	—
			1	—	Data
NO	2 bytes	0	0	Data1	Data2
YES				Data2	Data1

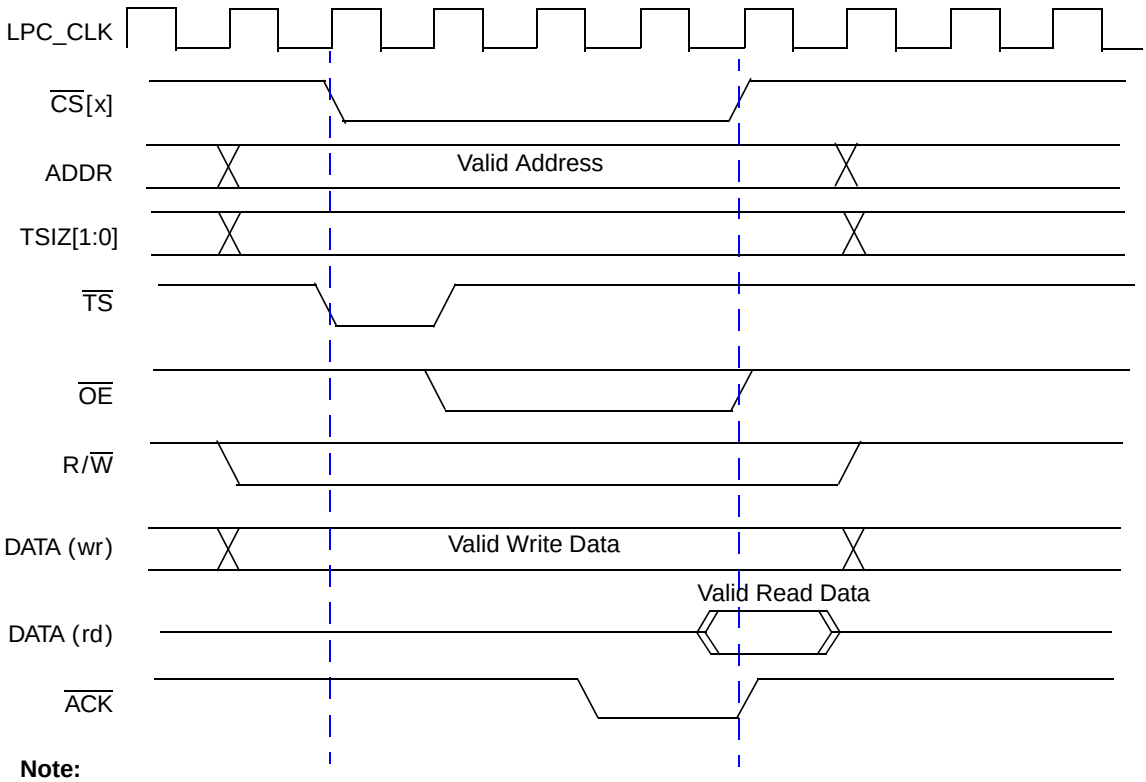
Note: Data1 is most significant byte and Data2 is lowest significant byte of data word.

1 Swap setting has no influence on byte transfers.

Figure 21-23 shows a non-muxed transaction. Figure 21-24 and Figure 21-25 show non-muxed synchronous bursts transactions. Figure 21-26 and Figure 21-27 show non-muxed asynchronous burst (page mode) transactions. Detailed information about timing diagrams and the influence of register settings can be found in the *MPC5125 Microcontroller Data Sheet*.

NOTE

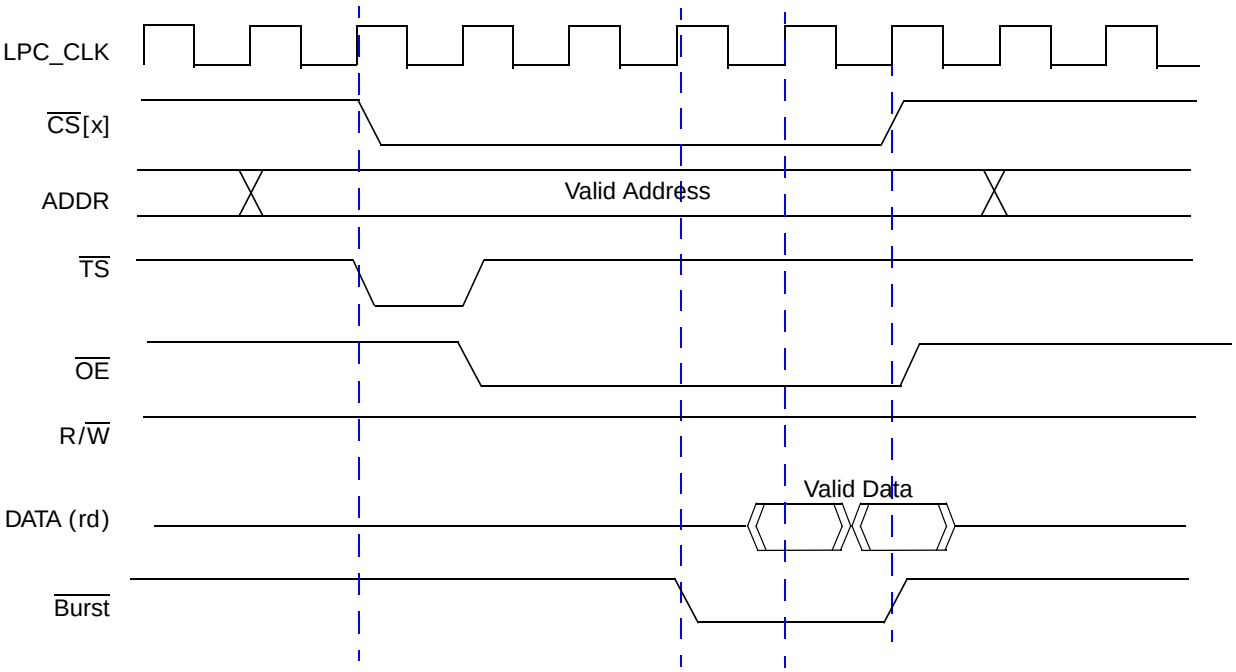
In the following five diagrams, deadcycle and holdcycle are each set to 0.



Note:

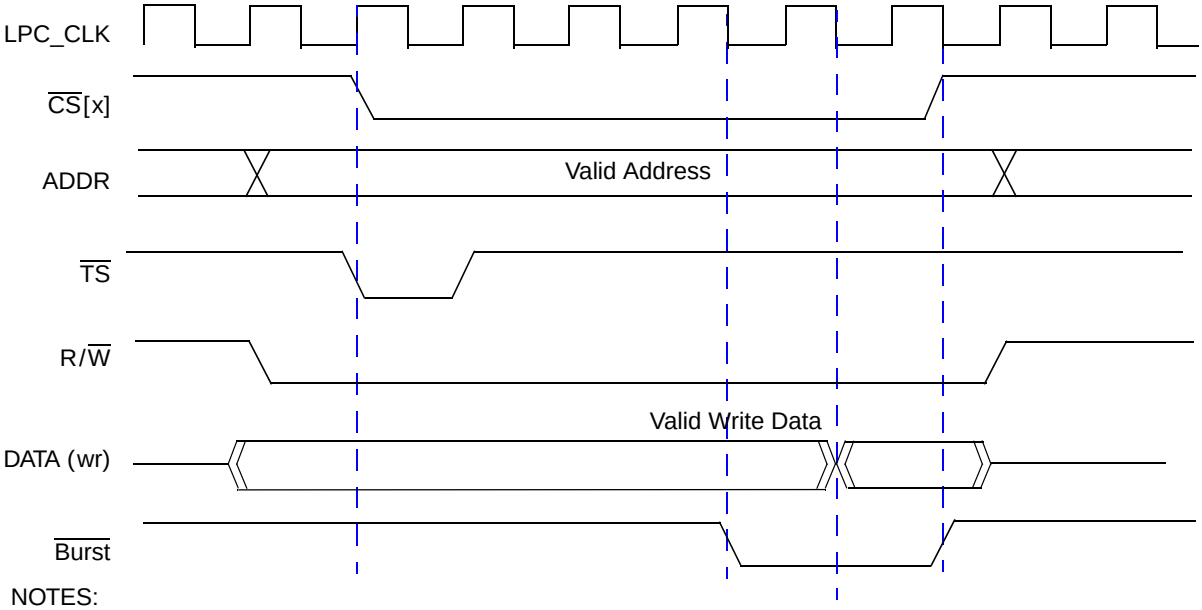
ACK can shorten the CS pulse width.
Related to the holdcycle setting, write data can stay on the bus longer.
This diagram represents a wait state setting of 2.

Figure 21-23. Timing Diagram—Non-Muxed Mode



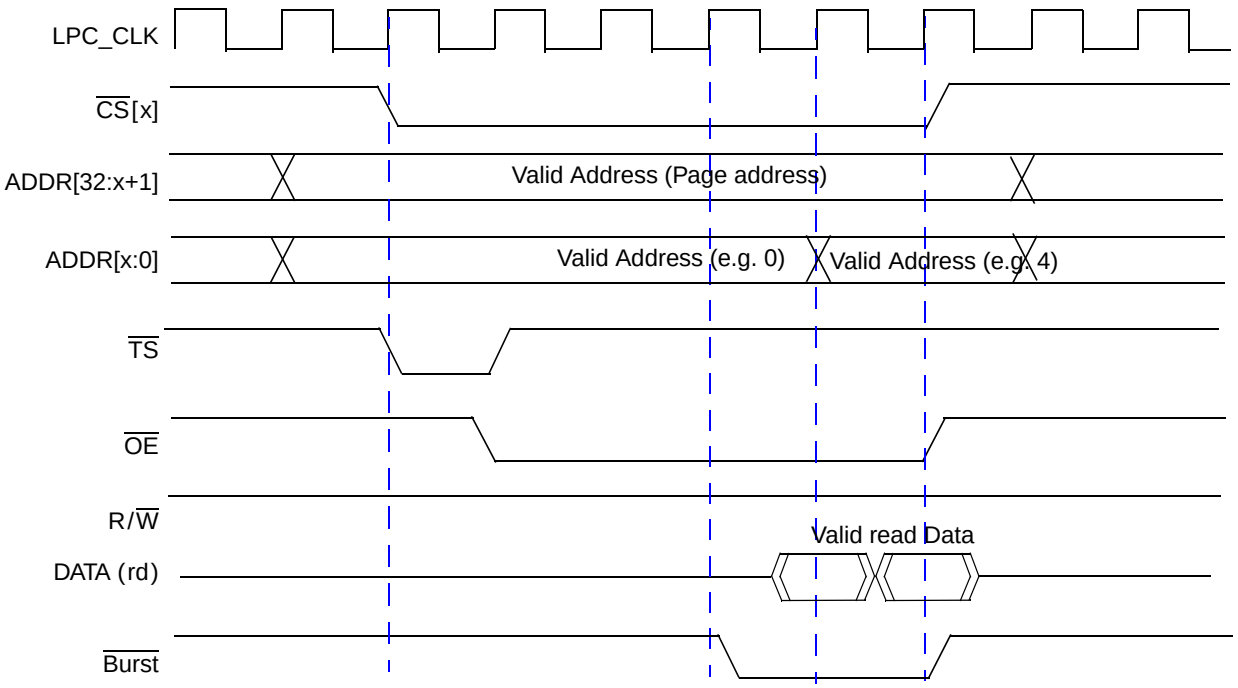
Note: This diagram represents a wait state setting of 1.

Figure 21-24. Timing Diagram—Non-Muxed Synchronous Read Burst



NOTES:
This diagram represents a wait state setting of 1.

Figure 21-25. Timing Diagram—Non-Muxed Synchronous Write Burst



Notes:

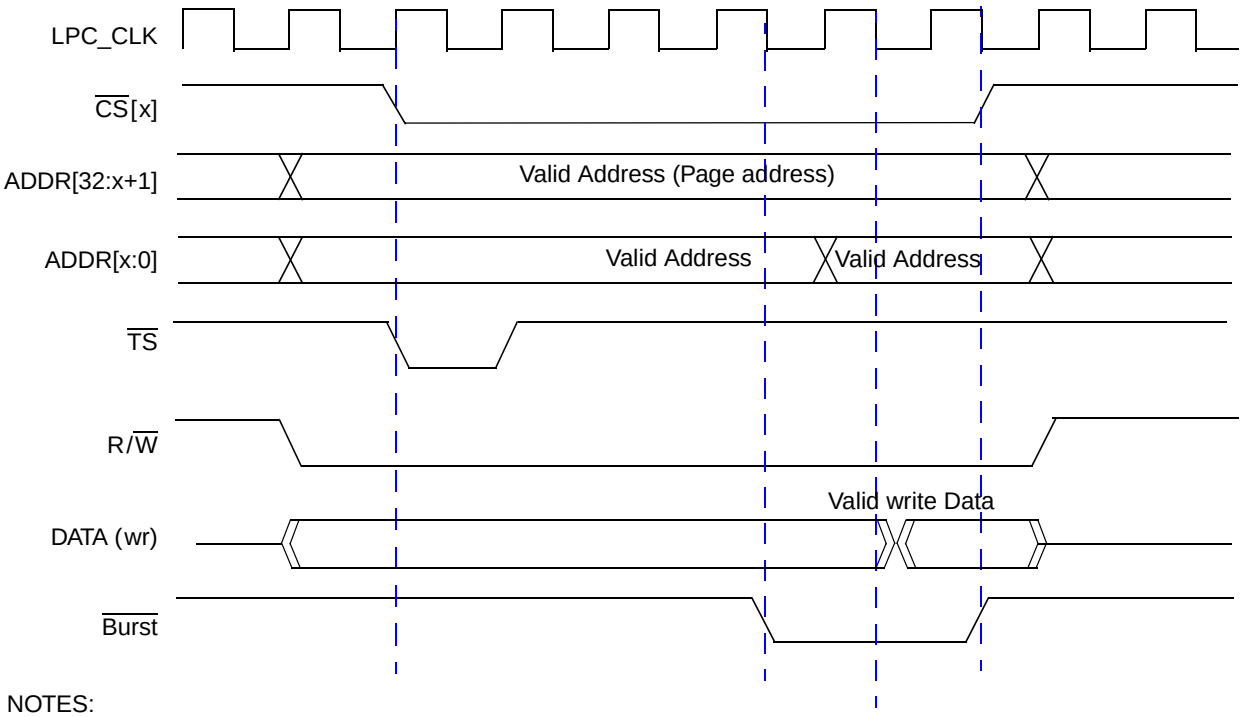
If the transaction is initiated by a master on the CSB, the address wraps around, if necessary, at cache level boundaries (e.g., 0x20; 0x40...)

If the transaction is initiated by the SCLPC module, the address continuously increases.

This diagram represents a wait state setting of 1.

The address changes at the rising edge of the LPC clock.

Figure 21-26. Timing Diagram—Non-Muxed Asynchronous Read Burst



NOTES:

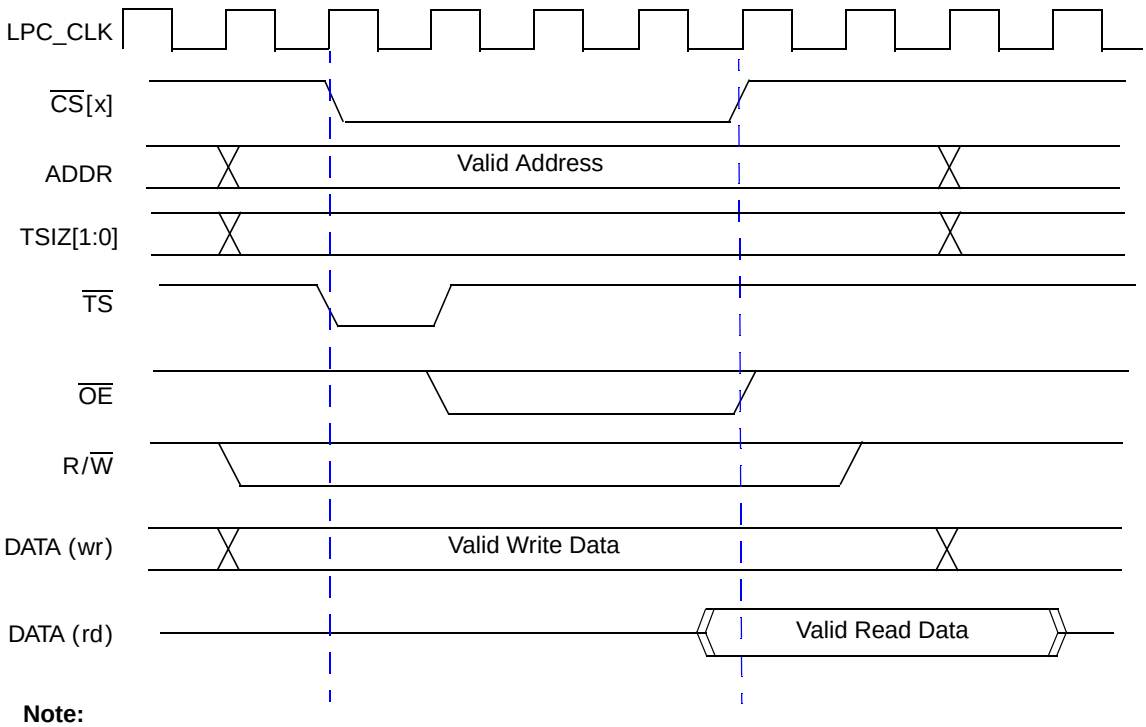
If the transaction is initiated by a master on the CSB, the address wraps around, if necessary, at cache level boundaries.

If the transaction is initiated by the SCLPC module, the address continuously increases with no address wrap-around.

This diagram represents a wait state setting of 1.

The address changes at the rising edge of the LPC clock.

Figure 21-27. Timing Diagram—Non-Muxed Asynchronous Write Burst



Note:

This diagram represents a holdcycle setting of 1.
This diagram represents a deadcycle setting of 2.
This diagram represents a wait state setting of 2.

Figure 21-28. Timing Diagram—Non-Muxed Mode with Different Timing Settings

In Non-Muxed mode the address and data are driven simultaneously on the external AD/AX bus. A single dedicated R/$\overline{W}$ pin is driven to indicate read or write. An individually dedicated CS pin is driven low while an external access is active.

Wait states are programmable and simply select how many LPC clocks the CS pin remain asserted. Separate values are available for read cycles versus write cycles. These values can be combined to create extremely long (up to 16 bits) cycles. Byte lane swapping is separately programmable between reads versus writes and can be used to perform endian conversions. The 24-bit data width is not supported.

Devices can be marked as read-only or write-only by setting a control bit in the appropriate LPC register. Attempted accesses in violation of this setting are prevented. Each CS pin can be individually enabled/disabled and the entire LPC module has a master enable bit. No software reset bit is provided or needed.

The non-muxed mode requires no external logic for interfacing to simple devices such as flash ROM, EEPROM or SRAM. It is faster than the muxed mode because data and address are provided in a single tenure.

21.3.2 Muxed Mode

In muxed mode, the addresses and data are muxed using dual tenure. First, the address is put on the shared address/data bus and $\overline{\text{ALE}}$ is asserted. The data is then driven when the chip select is asserted. Different modes of data sizes can be configured, as shown in Table 21-28.

Table 21-28. Muxed Mode Options

Data Size	Address Size	Comments
8	32	—
16	31	Short addressing; AD 0 is connected to internal address 1. Internal address 0 is not brought out.
16	32	Byte addressing
32	30	Word addressing; AD 0 is connected to internal address 2. Internal address 0 and 1 is not brought out.
32	32	Byte addressing

NOTE

The 24-bit data width is not supported.

Table 21-29. Internal Data and Address Bus Assignment to External Signals in Muxed Mode

Signal Name	8-Bit Data Bus			16-Bit Data Bus			32-Bit Data Bus		
	Addr Phase	Isolation	Data Phase	Addr Phase	Isolation	Data Phase	Addr Phase	Isolation	Data Phase
AX[03]	1	1	$\overline{\text{TS}}$	1	1	$\overline{\text{TS}}$	1	1	$\overline{\text{TS}}$
AX[02:01]	TSIZ[1:0]			TSIZ[1:0]			TSIZ[1:0]		
AX[00]	ALE[1]	!ALE	!ALE	ALE	!ALE	!ALE	ALE	!ALE	!ALE
AD[31:24]	address [31:24]	address [31:24]	0	address [31:24]	address [31:24]	0	address [31:24]	address [31:24]	data [31:24]
AD[23:16]	address [23:16]	address [23:16]	0	address [23:16]	address [23:16]	0	address [23:16]	address [23:16]	data [23:16]
AD[15:8]	address [15:8]	address [15:8]	0	address [15:8]	address [15:8]	data [15:8]	address [15:8]	address [15:8]	data [15:8]
AD[7:0]	address [7:0]	address [7:0]	data [7:0]	address [7:0]	address [7:0]	data [7:0]	address [7:0]	address [7:0]	data [7:0]

1 ALE represents the programmable value of the ALE bit (csboot/csx configuration registers).

Table 21-30. Internal Data and Address Bus Assignment to External Signals in Muxed Mode (Short/Word Addressing)

Signal Name	8-Bit Data Bus			16-Bit Data Bus			32-Bit Data Bus		
	Addr Phase	Isolation	Data Phase	Addr Phase	Isolation	Data Phase	Addr Phase	Isolation	Data Phase
AX[03]	1	1	TS	1	1	TS	1	1	TS
AX[02:01]	TSIZ[1:0]			TSIZ[1:0]			TSIZ[1:0]		
AX[00]	ALE[1]	!ALE	!ALE	ALE	!ALE	!ALE	ALE	!ALE	!ALE

Table 21-30. Internal Data and Address Bus Assignment to External Signals in Muxed Mode (Short/Word Addressing) (Continued)

Signal Name	8-Bit Data Bus			16-Bit Data Bus			32-Bit Data Bus		
	Addr Phase	Isolation	Data Phase	Addr Phase	Isolation	Data Phase	Addr Phase	Isolation	Data Phase
AD[31]	address 31	address 31	0	0	0	0	0	0	D31
AD[30]	address 30	address 30	0	address 31	address 31	0	0	0	D30
AD[29:24]	address [29:24]	address [29:24]	0	address [30:25]	address [30:25]	0	address [31:26]	address [31:26]	data [29:24]
AD[23:16]	address [23:16]	address [23:16]	0	address [24:17]	address [24:17]	0	address [25:18]	address [25:18]	data [23:16]
AD[15:8]	address [15:8]	address [15:8]	0	address [16:9]	address [16:9]	data [15:8]	address [17:10]	address [17:10]	data [15:8]
AD[7:0]	address [7:0]	address [7:0]	data [7:0]	address [8:1]	address [8:1]	data [7:0]	address [9:2]	address [9:2]	data [7:0]

1 ALE represents the programmable value of the ALE bit (csboot/csx configuration registers).

An ALE signal is asserted during this address tenure. ALE level is programmable to be low or high. The dedicated R/$\overline{W}$ output is also driven with ALE (and throughout the cycle). One clock after the ALE negates, the appropriate CS pin asserts (low) and the AD bus enters the data tenure (isolation cycle). The CS pin and this data tenure remain active until the programmed wait states expire or the device responds with an ACK assertion. ACK polarity is active low, but can be programmed to be ignored. The data tenure can contain up to the full 32-bit width. However, the data width is programmable to support dynamic bus-sized transactions.

The muxed mode requires external logic to latch the address during the address tenure and to decode bank selects if they are encoded. This mode is slower than the non-muxed mode because data and address are muxed in time. The supported address space is limited by the 30 address lines. In muxed mode, the LPC can access as much as 4 GB of data.

21.3.2.1 Address Tenure

The address is presented on the corresponding AD bus bits, which total 32 bits (AD[31:0]).

The TSIZ bits appear on AX[34] (TSIZ most significant bit) to AX[33] (TSIZ least significant bit). These bits are calculated and driven by the LPC based on the internal byte lane enables on the IP bus.

NOTE

Only TSIZes of one, two, or four are supported.

TSIZ [1:0] are driven as follows:

01 = Transaction is 1 byte

10 = Transaction is 2 bytes

00 = Transaction is 4 bytes

NOTE

Other values are invalid and should not be required by the external device.

Table 21-31 and Table 21-32 describe the various combinations of TSIZ, address, and byte lanes for 32-bit and 16-bit wide data buses. In addition, these tables show the influence of the WS and RS swap bits in the Chip Select X Configuration Register.

Table 21-31. Aligned Data Transfers for 32-Bit Data Bus Width

Swap WS or RS set	Transfer Size	TSIZ[1:0]	Address[1:0]	Data Lanes			
				AD[31:24]	AD[23:16]	AD[15:8]	AD[7:0]
NO/YES[1]	1 byte	01	00	Data	—	—	—
			01	—	Data	—	—
			10	—	—	Data	—
			11	—	—	—	Data
NO	2 bytes	10	00	Data1[2]	Data2	—	—
			10	—	—	Data1	Data2
YES			00	Data2	Data1	—	—
			10	—	—	Data2	Data1
NO	4 bytes	00	00	Data1	Data2	Data3	Data4
YES				Data4	Data3	Data2	Data1

1 Swap setting has no influence on byte transfers.

2 Data1 is the most significant byte and Data2 (2-byte transfer) or Data4 (4-byte transfer) is the least significant byte of the data word.

Table 21-32. Aligned Data Transfers for 16-Bit Data Bus Width

Swap WS or RS set	Transfer Size	TSIZ0	Address 0	Data Lanes	
				AD[15:8]	AD[7:0]
NO/YES[1]	1 byte	1	0	Data	—
			1	—	Data
NO	2 bytes	0	0	Data1[2]	Data2
YES				Data2	Data1

1 Swap setting has no influence on byte transfers.

2 Data1 is most significant byte and Data2 is lowest significant byte of data word.

The ALE signal is active low or high, depending on the ALE bit setting, and remains asserted for several external LPC bus clocks, depending on the ALEN bit field setting. Any external latch should be transparent when active.

21.3.2.2 Data Tenure

During data tenure, the following occurs:

- When a write to the device occurs, the LPC drives the indicated AD bits.
- When a read occurs, the indicated AD bits are tri-stated by the LPC.

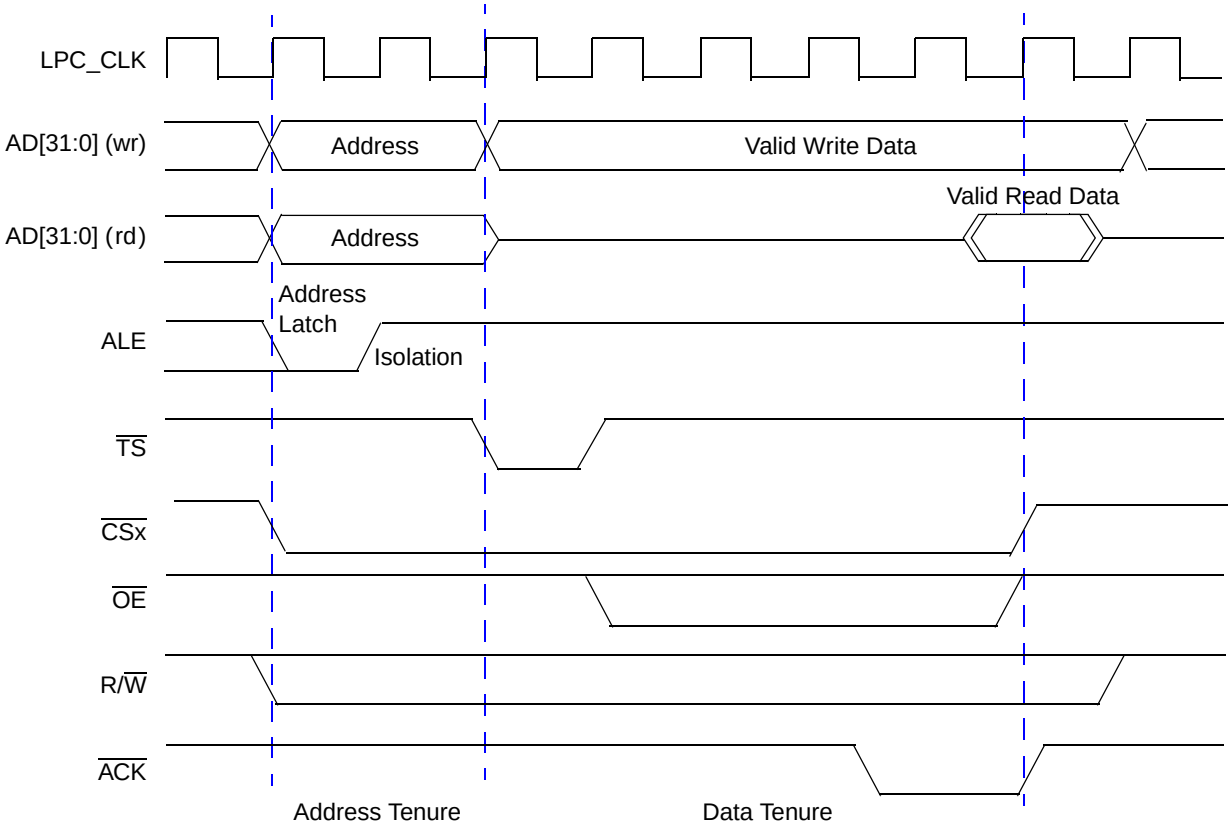
NOTE

AD[0] is treated as the least significant data bit. Any unused data bits (as indicated by the data size field in the associated control register) are driven low by the LPC. Therefore, they should not be driven by the device or glue chip.

The ACK input signal is internally synchronized to the internal clock. At the first LPC clock edge where the ACK input is detected as asserted, the LPC terminates the transaction and releases the bus on the next LPC bus clock. Figure 21-29 shows a muxed transaction-type timing diagram. Figure 21-30 and Figure 21-31 show muxed burst transactions. Detailed information about timing diagrams and the influence of register settings can be found in the *MPC5125 Microcontroller Data Sheet*.

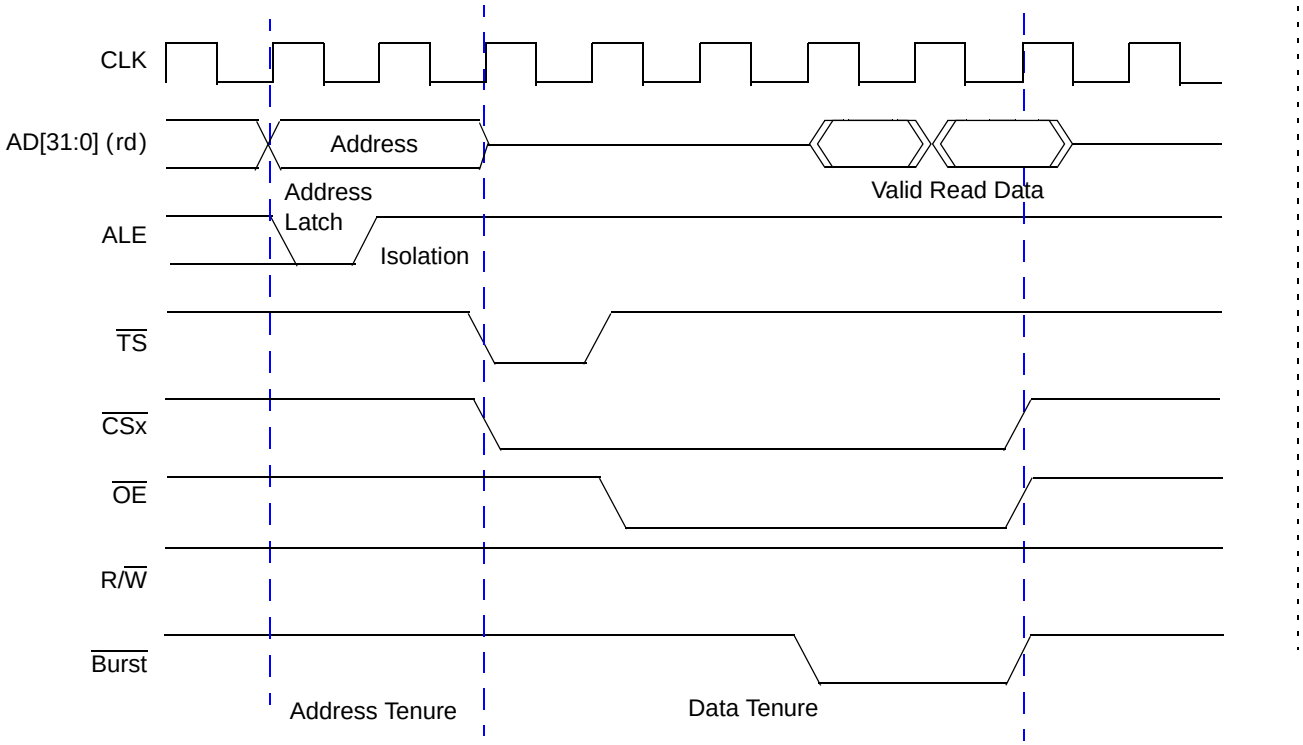
NOTE

In the following diagrams, deadcycle and holdcycle are each set to 0.



NOTES:
ACK can shorten the CS pulse width.
ALE can be active high or low, depending on the ale bit of the csboot/csx configuration register.
This diagram shows an active low scenario, which means the ALE bit is set to 0.
The length of the address latch and of the isolation cycle can be configured by the ALEN bit of the CSBOOT/CSX configuration register. This diagram shows a setting of 0.
This diagram represents a wait state setting of 3.
Address Latch bit are set to zero. CS is asserted together with ALE.

Figure 21-29. Timing Diagram – Muxed Mode



NOTES:

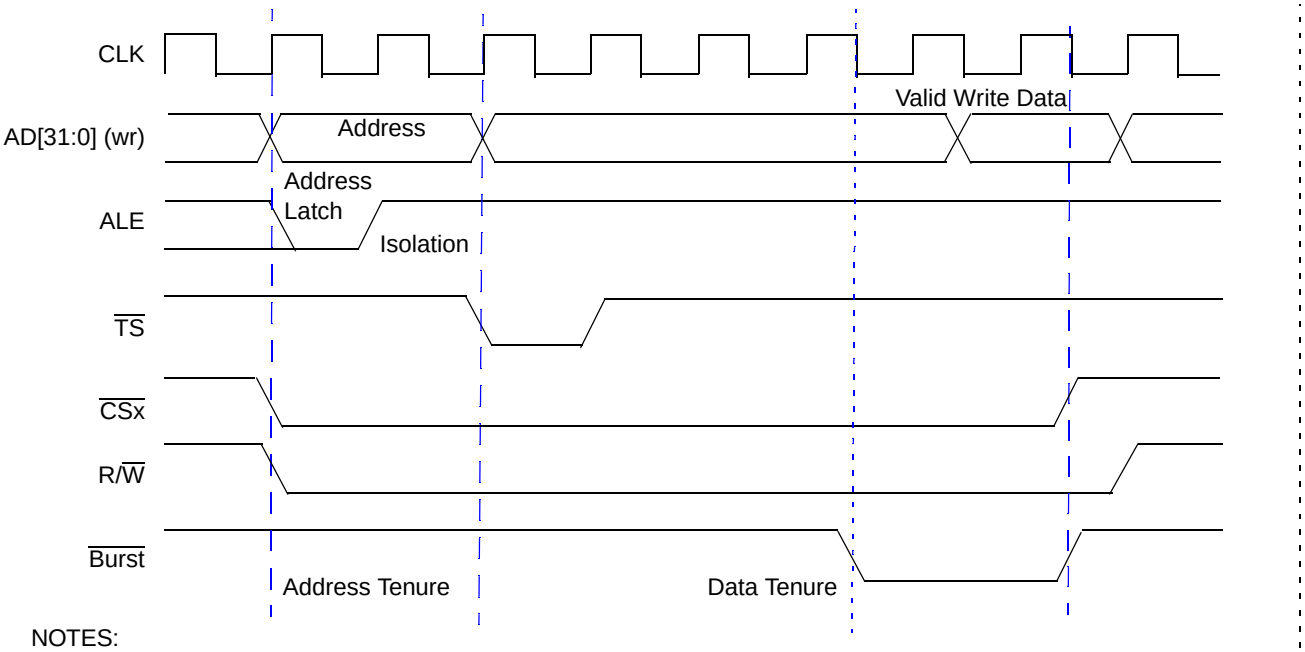
ALE can be active high or low. This diagram shows an active low scenario.

The length of the address latch and of the isolation cycle can be configured by the ALEN bit of the CSBOOT/CSX configuration register. This diagram shows a setting of 0.

This diagram represents a wait state setting of 1.

Address Latch bit are set to one. CS is asserted after the isolation cycle.

Figure 21-30. Timing Diagram – Muxed Synchronous Read Burst



NOTES:

ALE can be active high or low. This diagram shows an active low scenario.

The length of the address latch and of the isolation cycle can be configured by the ALEN bit of the CSBOOT/CSX configuration register. This diagram shows a setting of 0.

This diagram represents a wait state setting of 1.

Address Latch bit are set to zero. CS is asserted together with ALE.

Figure 21-31. Timing Diagram—Muxed Synchronous Write Burst

21.3.2.3 Boot Configuration

After reset, the core can fetch the first instruction from the LPC. The chip select boot (CSBoot) is dedicated for this purpose. CSBoot and CS0 are physically the same pins. The difference is that CSBoot is affected by the reset configuration and is enabled after reset.

Several options are available for this purpose:

- Muxed or non-muxed mode
- Byte or short word addressing
- Data size can be 8, 16, or 32 bits

21.3.2.4 Chip Selects Configuration

All chip selects (CS0–7) have the same functionality. Only one CS can be active at any given time. If an address hit is located in multiple CS windows, only the CS with the highest priority becomes active. The CS with the lowest number has the highest priority (CS0 = highest priority, CS7 = lowest priority).

CSBoot and CS0 are identical with the exception of their access window registers, contained in the XLBMEN register map (LocalPlus Boot/CS0–7 Access Window Registers (LPBAW/LPCS*x*AW)). CSBoot and CS0 are physically the same pins. The difference is that the CSBoot access window is affected by the reset configuration and is the only enabled chip select after reset.

To execute code from CS0 instead of CSBoot, the CS0 access window register must be configured and a jump instruction into this address space needs to be executed.

Deadcycles from 0 to 3 can be added to any CS read access and occur in addition to any cycles that already exist. The chip select deadcycle control register configures deadcycles.

Holdcycles from 0 to 3 can be added to any CS write access and occur in addition to any cycles that already exist.

Burst mode operations are supported on all CSs and all modes and can be configured by the chip select burst control register.

The following features are valid for all CSs and for both modes (non-muxed and muxed):

- Supports 8-, 16-, and 32-bit data bus width
- Supports dynamic bus sizing, which means read and write transactions greater than the defined port size are possible
- Transactions less than the defined port size are supported only if the device can decode the Tsiz[1:0] bits, which indicate the current transaction size.
- Supports code execution
  - The e300 processor can execute code from the LP bus from all CS
- Supports burst access (read and write)
  - Synchronous burst

Asynchronous burst mode (Page mode) is supported in non-muxed mode.

21.3.3 SCLPC Interface

The SCLPC interface, together with the DMA engine, can be used to transfer data between an external device and the DRAM without the usage of the e300 processor.

SCLPC controls the data transfer between the LPC RX/TX FIFO and the external device. The external transfer behavior is defined by the CSBoot/CS[x] configuration, chip select burst control, chip select deadcycle, and chip select holdcycle registers. The supported transfer sizes are limited, by BPT bit field setting, to 1, 2, 4, 8, 16, 24, 32, 40, 48, or 56 bytes only. Burst transfers could be generated with transfer size settings greater as 4.

The SCLPC controller supports half-duplex operation (transmit or receive) only. If software configures a transmit packet, the packet must be complete before a receive operation can be configured and started. The length of one transfer packet is defined with the Packet Size bit field of the SCLPC Packet Size (LPC_SCLPC_PS) Register. A transfer packet is split down into smaller bytes-per-transfer (BPT) slice. A BPT slice cannot be interrupted by a direct e300 access to an external device. A BPT slice needs to be finished first before the e300 access goes onto the external bus.

21.3.3.1 SCLPC Programming

The device-specific behavior (non-/muxed, burst, wait cycles—Table 21-3 through Table 21-11) needs to be programmed before the SCLPC is set up.

The following steps set up the SCLPC controller together with LPC RX/TX FIFO:

1. Configure DMA to transfer data between LPC RX/TX FIFO and DRAM.

   The maximum Inner Minor Byte Count setting, working together with the LPC RX/TX FIFO, is 32 bytes
2. Configure the granularity watermark to 28. This is done by setting the GR bit field (see Section 21.2.1.3.3, “LPC RX/TX FIFO Control (LPC_SCLPC_FIFOC) Register”) to 7.
3. Configure the alarm watermark (ALARM_W) bit field (see Section 21.2.1.3.4, “LPC RX/TX FIFO Alarm (LPC_SCLPC_FIFOA) Register”). The setting is depending on the system load.
4. Configure start address, chip select, direction (read/write), address increment, and bytes per transfer. In case of a read transfer from an external device, set the Flush bit. This bit does not need to be set for a write transfer.
5. Set the packet size. Do not set the Restart bit yet.
6. Enable the SCLPC interrupt by setting the ME and NIE bits of the SCLPC Enable (LPC_SCLPC_E) Register.
7. Reset the SCLPC controller and the FIFO.
8. Enable the SCLPC controller and the FIFO by clearing the SCLPC controller and FIFO reset bits.

The following steps start SCLPC transfers:

1. Set Restart bit of SCLPC Packet Size (LPC_SCLPC_PS) Register.

   In case of a read transaction, the SCLPC controller starts to fill the LPC RX/TX FIFO.

   In case of a write transaction, the SCLPC controller is waiting until BPT size data is written by the DMA into the LPC RX/TX FIFO.
2. Start the LPC DMA request within the DMA module.

The transfer is finished when

- In case of a read transfer, the DMA signals via interrupt that the LPC TCD is finished
- In case of a write transfer, the SCLPC signals via interrupt that the transfer is finished

21.3.4 Programmer’s Model

Table 21-3 through Table 21-11 describe the registers and bit meanings for configuring CS operation in detail. There are eight identical CS configuration registers, one for each CS output. However, the CSBoot ROM configuration register has active defaults for use by BOOTROM on CS0. All other configuration registers are disabled at powerup and require software intervention before the corresponding CS operates. The chip select control register is the enable register and the chip select status register serves as a status register. The chip select burst control register enables burst mode and deadcycles are configured by the chip select deadcycle control register.

NOTE

The address range registers for each CS reside in the XLBMEN register set rather than in the LPC register set. See Section 21.3.2.3, “Boot Configuration.”

Chapter 22
MSCAN

22.1 Introduction

This module contains four identical and independent MSCAN controllers.

The module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the BOSCH specification dated September 1991. To fully understand the MSCAN specification, read the Bosch specification first to familiarize yourself with terms and concepts contained within this document.

The CAN protocol was primarily designed as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth.

MSCAN uses an advanced buffer arrangement resulting in a predictable real-time behavior and simplifies the application software.

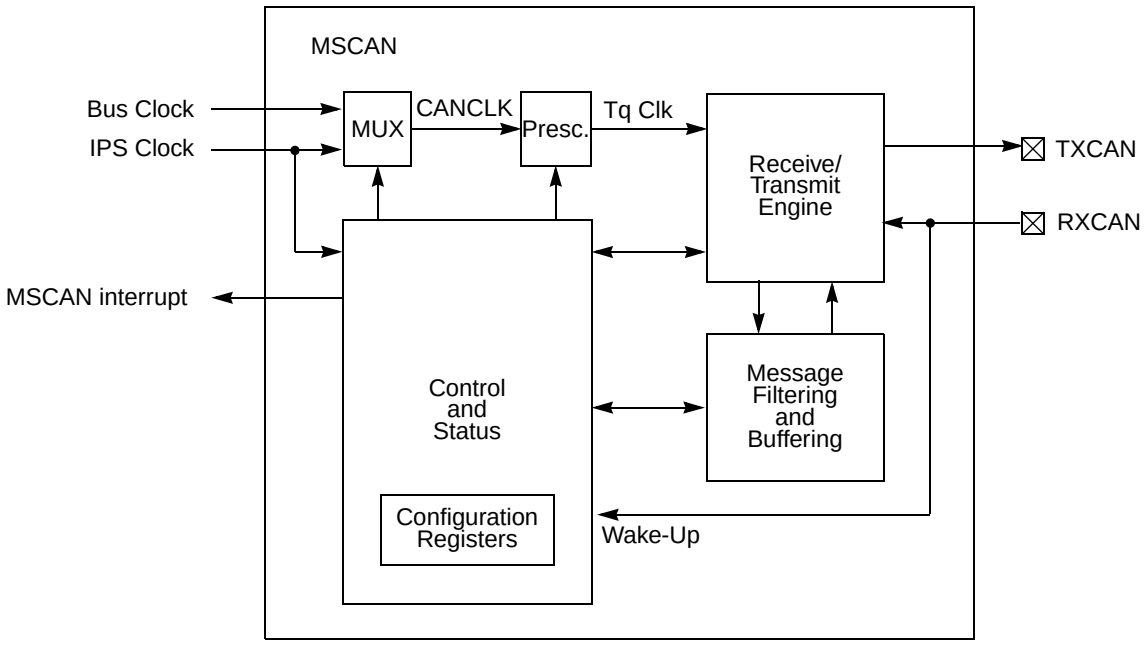


Figure 22-1. MSCAN Block Diagram

22.1.1 Features

The basic features of the MSCAN are:

- Implementation of the CAN protocol – Version 2.0A/B

- Standard and extended data frames
- 0–8 bytes data length
- Programmable bit rate as fast as 1 Mbit/s, depending on actual bit timing and clock jitter of PLL
- Support for remote frames
- Five receive buffers with FIFO storage scheme

- Three transmit buffers with internal prioritization using a local priority concept
- Flexible maskable identifier filter supports two full-size extended identifier filters (two 32-bit) or four 16-bit filters or eight 8-bit filters
- Programmable wake-up functionality
- Programmable loop back mode supports self-test operation
- Programmable listen-only mode for monitoring of CAN bus
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (warning, error passive, bus-off)
- Programmable clock source. See Chapter 5, “Clocks and Low-Power Modes.”
- Internal timer for time-stamping of received and transmitted messages
- Three low power modes: sleep, power down and MSCAN enable
- Programmable bus-off recovery functionality
- Global initialization of configuration registers

22.2 External Signal Description

The MSCAN uses two external pins. In the module, the MSCAN pins are shared with other functionary and can be available at four different groups of pins. The configuration of the pin-muxing is controlled by the Port Configuration Register.

22.2.1 CAN Receiver Input Pins

Table 22-1. Signal Properties

Name	Port	Function	I/O
CAN1_RX	Input	MSCAN1 receiver input pin	I
CAN1_TX/ PSC0_4	Output	MSCAN1 transmitter output pin. The CAN1_TX/PSC0_4 output pin represents the logic level on the CAN bus: 0 Dominant state 1 Recessive state	O
CAN2_RX	Input	MSCAN2 receiver input pin	I
CAN2_TX/ PSC1_0	Output	MSCAN2 transmitter output pin. The CAN2_TX/PSC1_0 output pin represents the logic level on the CAN bus: 0 Dominant state 1 Recessive state	O
DIU_LD00/ I2C1_SCL	Input	MSCAN3 receiver input pin	I
DIU_LD01/ I2C1_SDA	Output	MSCAN3 transmitter output pin. The DIU_LD01/I2C1_SDA output pin represents the logic level on the CAN bus: 0 Dominant state 1 Recessive state	O
DIU_LD08	Input	MSCAN4 receiver input pin	I
DIU_LD09	Output	MSCAN4 transmitter output pin. The DIU_LD09 output pin represents the logic level on the CAN bus: 0 Dominant state 1 Recessive state	O

CAN1_RX, CAN2_RX, DIU_LD00, I2C1_SCL, and DIU_LD08 are the MSCAN receiver input pins. MSCAN3 can be configured to select DIU_LD00 or I2C1_SCL as input pin.

22.2.2 CAN Transmitter Output Pins

CAN1_TX, CAN2_TX, DIU_LD01, I2C1_SDA, and DIU_LD09 are MSCAN transmitter output pins. MSCAN3 can be configured to select DIU_LD01 or I2C1_SDA as output pin. These pins represents the logic level on the CAN bus:

0 = Dominant state

1 = Recessive state

22.2.3 CAN System

A typical CAN system with MSCAN is shown in Figure 22-2. Each CAN station is connected physically to the CAN bus lines through a transceiver device. The transceiver is capable of driving the large current needed for the CAN bus and has current protection against defective CAN or defective stations.

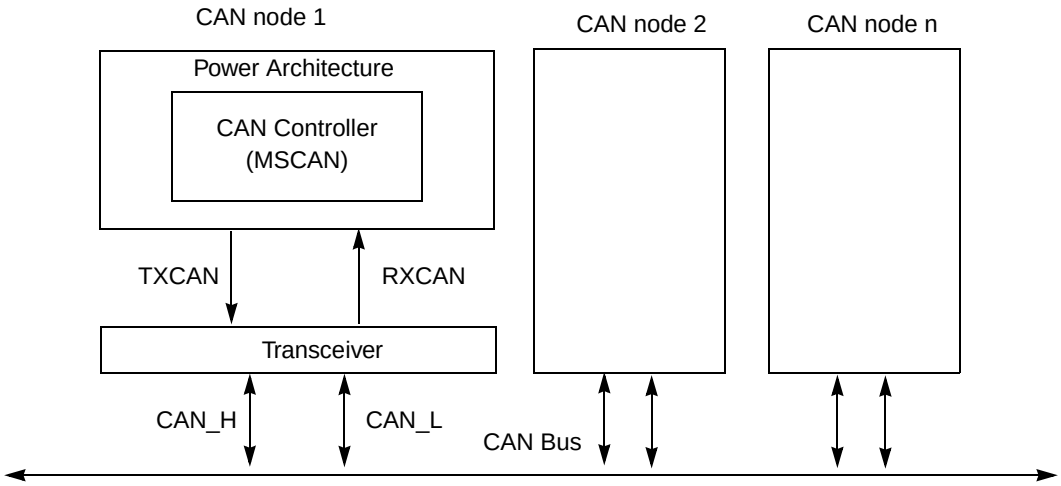


Figure 22-2. MSCAN System

22.3 Memory Map and Register Definition

The MPC5125 contains four independent MSCAN Controllers with identical register sets. The register sets have different base addresses. However, the registers in each set have the same offsets with respect to their base addresses.

Table 22-2. FSL memory map

Offset from MSCAN_BASE[1] MSCAN1: 0xFF40_1300 MSCAN2: 0xFF40_1380 MSCAN3: 0xFF40_2300 MSCAN4: 0xFF40_2380	Register	Access[2]	Reset Value	Section/Page
0x00	MSCAN Control Register 0 (CANCTL0)	R/W	0x01	22.3.2.1/22-567
0x01	MSCAN Control Register 1 (CANCTL1)	R/W	0x11	22.3.2.2/22-569
0x02–0x03	Reserved			
0x04	MSCAN Bus Timing Register 0 (CANBTR0)	R/W	0x00	22.3.2.3/22-570
0x05	MSCAN Bus Timing Register 1 (CANBTR1)	R/W	0x00	22.3.2.4/22-571
0x06–0x07	Reserved			
0x08	MSCAN Receiver Flag Register (CANRFLG)	R/W	0x00	22.3.2.5/22-572
0x09	MSCAN Receiver Interrupt Enable Register (CANRIER)	R/W	0x00	22.3.2.6/22-573
0x0A–0x0B	Reserved			
0x0C	MSCAN Transmitter Flag Register (CANTFLG)	R/W	0x07	22.3.2.7/22-575
0x0D	MSCAN Transmitter Interrupt Enable Register (CANTIER)	R/W	0x00	22.3.2.8/22-575
0x0E–0x0F	Reserved			
0x10	MSCAN Transmitter Message Abort Control (CANTARQ)	R/W	0x00	22.3.2.9/22-576

Table 22-2. FSL memory map (Continued)

Offset from MSCAN_BASE[1] MSCAN1: 0xFF40_1300 MSCAN2: 0xFF40_1380 MSCAN3: 0xFF40_2300 MSCAN4: 0xFF40_2380	Register	Access[2]	Reset Value	Section/Page
0x11	MSCAN Transmitter Message Abort Control (CANTAAK)	R	0x00	22.3.2.10/22-577
0x12–0x13	Reserved			
0x14	MSCAN Transmit Buffer Selection (CANTBSEL)	R/W	0x00	22.3.2.11/22-578
0x15	MSCAN Identifier Acceptance Control Register (CANIDAC)	R/W	0x00	22.3.2.12/22-579
0x16–0x18	Reserved			
0x19	MSCAN MISC Register (CANMISC)	R/W	0x00	22.3.2.13/22-579
0x1A–0x1B	Reserved			
0x1C	MSCAN Receive Error Counter Register (CANRXERR)	R	0x00	22.3.2.14/22-580
0x1D	MSCAN Transmitter Error Counter Register (CANTXERR)	R	0x00	22.3.2.15/22-580
0x1E–0x1F	Reserved			
0x20	MSCAN Identifier Acceptance Registers (CANIDAR0)	R/W	0x00	22.3.2.16/22-581
0x21	MSCAN Identifier Acceptance Registers (CANIDAR1)	R/W	0x00	22.3.2.16/22-581
0x22–0x23	Reserved			
0x24	MSCAN Identifier Acceptance Registers (CANIDAR2)	R/W	0x00	22.3.2.16/22-581
0x25	MSCAN Identifier Acceptance Registers (CANIDAR3)	R/W	0x00	22.3.2.16/22-581
0x26–0x27	Reserved			
0x28	MSCAN Identifier Mask Registers (CANIDMR0)	R/W	0x00	22.3.2.17/22-582
0x29	MSCAN Identifier Mask Registers (CANIDMR1)	R/W	0x00	22.3.2.17/22-582
0x2A–0x2B	Reserved			
0x2C	MSCAN Identifier Mask Registers (CANIDMR2)	R/W	0x00	22.3.2.17/22-582
0x2D	MSCAN Identifier Mask Registers (CANIDMR3)	R/W	0x00	22.3.2.17/22-582

Table 22-2. FSL memory map (Continued)

Offset from MSCAN_BASE[1] MSCAN1: 0xFF40_1300 MSCAN2: 0xFF40_1380 MSCAN3: 0xFF40_2300 MSCAN4: 0xFF40_2380	Register	Access[2]	Reset Value	Section/Page
0x2E–0x2F	Reserved			
0x30	MSCAN Identifier Acceptance Registers (CANIDAR4)	R/W	0x00	22.3.2.16/22-581
0x31	MSCAN Identifier Acceptance Registers (CANIDAR5)	R/W	0x00	22.3.2.16/22-581
0x32–0x33	Reserved			
0x34	MSCAN Identifier Acceptance Registers (CANIDAR6)	R/W	0x00	22.3.2.16/22-581
0x35	MSCAN Identifier Acceptance Registers (CANIDAR7)	R/W	0x00	22.3.2.16/22-581
0x36–0x37	Reserved			
0x38	MSCAN Identifier Mask Registers (CANIDMR4)	R/W	0x00	22.3.2.17/22-582
0x39	MSCAN Identifier Mask Registers (CANIDMR5)	R/W	0x00	22.3.2.17/22-582
0x3A–0x3B	Reserved			
0x3C	MSCAN Identifier Mask Registers (CANIDMR6)	R/W	0x00	22.3.2.17/22-582
0x3D	MSCAN Identifier Mask Registers (CANIDMR7)	R/W	0x00	22.3.2.17/22-582
0x3E–0x3F	Reserved			
0x40–0x5F	**MSCAN Receive Message Buffer (CANRXFG)[3]**			22.4.3/22-595
0x40	RX Identifier Register 0 (IDR0)	R/W	—[4]	22.3.3.1/22-585
0x41	RX Identifier Register 1 (IDR1)	R/W	—[4]	22.3.3.1/22-585
0x42–0x43	Reserved			
0x44	RX Identifier Register 2 (IDR2)	R/W	—[4]	22.3.3.1/22-585
0x45	RX Identifier Register 3 (IDR3)	R/W	—[4]	22.3.3.1/22-585
0x46–0x47	Reserved			
0x48	RX Data Segment Register 0 (DSR0)	R/W	—[4]	22.3.3.2/22-588
0x49	RX Data Segment Register 1 (DSR1)	R/W	—[4]	22.3.3.2/22-588
0x4A–0x4B	Reserved			
0x4C	RX Data Segment Register 2 (DSR2)	R/W	—[4]	22.3.3.2/22-588
0x4D	RX Data Segment Register 3 (DSR3)	R/W	—[4]	22.3.3.2/22-588

Table 22-2. FSL memory map (Continued)

Offset from MSCAN_BASE[1] MSCAN1: 0xFF40_1300 MSCAN2: 0xFF40_1380 MSCAN3: 0xFF40_2300 MSCAN4: 0xFF40_2380	Register	Access[2]	Reset Value	Section/Page
0x4E–0x4F	Reserved			
0x50	RX Data Segment Register 4 (DSR4)	R/W	—[4]	22.3.3.2/22-588
0x51	RX Data Segment Register 5 (DSR5)	R/W	—[4]	22.3.3.2/22-588
0x52–0x53	Reserved			
0x54	RX Data Segment Register 6 (DSR6)	R/W	—[4]	22.3.3.2/22-588
0x55	RX Data Segment Register 7 (DSR7)	R/W	—[4]	22.3.3.2/22-588
0x56–0x57	Reserved			
0x58	RX Data Length Register (DLR)	R/W	—[4]	22.3.3.3/22-589
0x59	RX Transmit Buffer Priority Register (TBPR)	R/W	0x00	22.3.3.4/22-589
0x5A–0x5B	Reserved			
0x5C	RX Time Stamp Register High (TSRH)	R	—[4]	22.3.3.5/22-590
0x5D	RX Time Stamp Register Low (TSRL)	R	—[4]	22.3.3.5/22-590
0x5E–0x5F	Reserved			
0x60–0x7F	**MSCAN Transmit Message Buffer (CANTXFG)[5]**			22.4.3/22-595
0x60	TX Identifier Register 0 (IDR0)	R/W	—[4]	22.3.3.1/22-585
0x61	TX Identifier Register 1 (IDR1)	R/W	—[4]	22.3.3.1/22-585
0x62–0x63	Reserved			
0x64	TX Identifier Register 2 (IDR2)	R/W	—[4]	22.3.3.1/22-585
0x65	TX Identifier Register 3 (IDR3)	R/W	—[4]	22.3.3.1/22-585
0x66–0x67	Reserved			
0x68	TX Data Segment Register 0 (DSR0)	R/W	—[4]	22.3.3.2/22-588
0x69	TX Data Segment Register 1 (DSR1)	R/W	—[4]	22.3.3.2/22-588
0x6A–0x6B	Reserved			
0x6C	TX Data Segment Register 2 (DSR2)	R/W	—[4]	22.3.3.2/22-588
0x6D	TX Data Segment Register 3 (DSR3)	R/W	—[4]	22.3.3.2/22-588
0x6E–0x6F	Reserved			
0x70	TX Data Segment Register 4 (DSR4)	R/W	—[4]	22.3.3.2/22-588
0x71	TX Data Segment Register 5 (DSR5)	R/W	—[4]	22.3.3.2/22-588
0x72–0x73	Reserved			

Table 22-2. FSL memory map (Continued)

Offset from MSCAN_BASE[1] MSCAN1: 0xFF40_1300 MSCAN2: 0xFF40_1380 MSCAN3: 0xFF40_2300 MSCAN4: 0xFF40_2380	Register	Access[2]	Reset Value	Section/Page
0x74	TX Data Segment Register 6 (DSR6)	R/W	—[4]	22.3.3.2/22-588
0x75	TX Data Segment Register 7 (DSR7)	R/W	—[4]	22.3.3.2/22-588
0x76–0x77	Reserved			
0x78	TX Data Length Register (DLR)	R/W	—[4]	22.3.3.3/22-589
0x79	TX Transmit Buffer Priority Register (TBPR)	R/W	0x00	22.3.3.4/22-589
0x7A–0x7B	Reserved			
0x7C	TX Time Stamp Register High (TSRH)	R	—[4]	22.3.3.5/22-590
0x7D	TX Time Stamp Register Low (TSRL)	R	—[4]	22.3.3.5/22-590
0x7E–0x7F	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, "System Configuration and Memory Map (XLBMEN + Mem Map)."

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

3 Reserved bits and unused bits within the RX-buffer (CANRXFG) are read as x, because of RAM-based implementation.

4 Reset value is indeterminate.

5 Reserved bits and unused bits within the TX-buffer (CANTXFG) are read as x, because of RAM-based implementation.

22.3.1 Register Summary

Table 22-3 shows a summary of the registers.

Table 22-3. Register Summary

Name		7	6	5	4	3	2	1	0
0x00 CANCTL0	R	RXFRM	RXACT		SYNCH	TIME	WUPE	SLPRQ	INITRQ
	W								
0x01 CANCTL1	R	CANE	CLKSRC	LOOPB	LISTEN	0		SLPAK	INITAK
	W								
0x04 CANBTR0	R	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
	W								
0x05 CANBTR1	R	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
	W								
0x08 CANRFLG	R	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
	W								

Table 22-3. Register Summary (Continued)

Name		7	6	5	4	3	2	1	0
0x09 CANRIER	R	WUPIE	CSCIE	RSTATE1	RSTATE0	TSTATE1	TSTATE0	OVRIE	RXFIE
	W								
0x0C CANTFLG	R	0	0	0	0	0	TXE2	TXE1	TXE0
	W								
0x0D CANTIER	R	0	0	0	0	0	TXEIE2	TXEIE1	TXEIE0
	W								
0x10 CANTARQ	R	0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
	W								
0x11 CANTAAK	R	0	0	0	0	0	ABTAK2	ABTAK1	ABTAK0
	W								
0x14 CANTBSEL	R	0	0	0	0	0	TX2	TX1	TX0
	W								
0x15 CANIDAC	R	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
	W								
0x19 CANMISC	R	0	0	0	0	0	0	0	BOFFHOLD
	W								
0x1C CANRXERR	R	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
	W								
0x1D CANTXERR	R	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
	W								
0x20 CANIDAR0	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
0x21 CANIDAR1	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
0x24 CANIDAR2	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
0x25 CANIDAR3	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
0x28 CANIDMR0	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
0x29 CANIDMR1	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
0x2C CANIDMR2	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
0x2D CANIDMR3	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
0x30 CANIDAR4	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								

Table 22-3. Register Summary (Continued)

Name		7	6	5	4	3	2	1	0
0x31 CANIDAR5	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
0x34 CANIDAR6	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
0x35 CANIDAR7	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
0x38 CANIDMR4	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
0x39 CANIDMR5	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
0x3C CANIDMR6	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
0x3D CANIDMR7	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
0x40-0x5F CANRXFG	R	FOREGROUND RECEIVE BUFFER							
	W								
0x60-7F CANTXFG	R	FOREGROUND TRANSMIT BUFFER							
	W								

22.3.2 Register Descriptions

This section describes in detail all the registers and register bits in the MSCAN module. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagram, in bit order. All bits of all registers in this module are completely synchronous to internal clocks during a register read.

22.3.2.1 MSCAN Control 0 Register (CANCTL0)

This register provides for various control of the MSCAN module as described below.

Read: Anytime

Write: Anytime when out of initialization mode; exceptions are read-only RXACT and SYNCH, RXFRM (which is set by the module only), and INITRQ (which is also writable in initialization mode).

NOTE

The CANCTL0 register (except the WUPE, INITRQ, and SLPRQ bits) is held in the reset state when initialization mode is active (INITRQ = 1 and INITAK = 1). The CSWAI, TIME, WUPE, and SLPRQ bits are writable as soon as the initialization mode is complete (INITRQ = 0 and INITAK = 0).

Address: Base + 0x00 Access: User read/write

	0	1	2	3	4	5	6	7
R	RXFRM	RXACT	0	SYNCH	TIME	WUPE	SLPRQ	INITRQ
W								
Reset	0	0	0	0	0	0	0	1

Figure 22-3. MSCAN Control 0 Register (CANCTL0)

Table 22-4. CANCTL0 field descriptions

Field	Description
RXFRM[1]	Received Frame Flag bit is read and clear only. It is set when a receiver has received a valid message correctly, independently of the filter configuration. After set, it remains set until cleared by software or reset. Clear by writing 1 to the bit. This bit is not valid in loop-back mode. 0 No valid message was received since last clearing this flag. 1 A valid message was received since last clearing of this flag.
RXACT	Receiver Active Status bit indicates MSCAN is receiving a message. The receiver front end controls the flag. This bit is not valid in loop-back mode. 0 MSCAN is transmitting or idle[2]. 1 MSCAN is receiving a message (including when arbitration is lost).
SYNCH	Synchronized Status bit indicates whether MSCAN is synchronized to the CAN bus and can participate in the communication process. It is set and cleared by MSCAN. 0 MSCAN is not synchronized to the CAN bus. 1 MSCAN is synchronized to the CAN bus.

Table 22-4. CANCTL0 field descriptions (Continued)

Field	Description
TIME	Timer Enable bit activates an internal 16-bit wide free running timer clocked by the bit-clock. If timer is enabled, a 16-bit time stamp is assigned to each transmitted/received message within the active Tx/Rx buffer. As soon as a message is acknowledged on CAN, the time stamp is written to the highest bytes (0x1C, 0x1D) in the appropriate buffer (see Section 22.3.3, “Programmer’s Model of Message Storage”). The internal timer is reset (all bits set to 0) when initialization mode is active. 0 Disable internal MSCAN timer. 1 Enable internal MSCAN timer.
WUPE[3]	Wake-Up Enable bit lets MSCAN restart when being locked in idle state during sleep mode and traffic on CAN is detected (see Section 22.4.8.1, “MSCAN Sleep Mode”). 0 Wake-Up disabled. The MSCAN ignores traffic on CAN. 1 Wake-Up enabled.The MSCAN is able to restart.
SLPRQ[4]	Sleep Mode Request bit requests MSCAN enter sleep mode, an internal power saving mode (see Section 22.4.8.1, “MSCAN Sleep Mode”). The sleep mode request is serviced when the CAN bus is idle, i.e., the module is not receiving a message and all transmit buffers are empty. The module indicates entry to sleep mode by setting SLPAK = 1 (see Section 22.3.2.2, “MSCAN Control 1 Register (CANCTL1)”). Sleep mode is active until SLPRQ is cleared by the Power Architecture or, depending on the setting of WUPE, the MSCAN detects activity on the CAN bus and clears SLPRQ itself. 0 Running.The MSCAN functions normally. 1 Sleep Mode Request. The MSCAN locks in idle state.
INITRQ[5,6]	Initialization Mode Request. When this bit is set by the Power Architecture, the MSCAN skips to initialization mode (see Section 22.4.8.2, “MSCAN Initialization Mode”). Any ongoing transmission or reception is aborted and synchronization to the CAN bus is lost. The module indicates entry to initialization mode by setting INITAK = 1 (Section 22.3.2.2, “MSCAN Control 1 Register (CANCTL1)”). The following registers enter their hard reset state and restore their default values: CANCTL0[7], CANRFLG[8], CANRIER[9], CANTFLG, CANTIER, CANTARQ, CANTAAK, and CANTBSEL. The registers CANCTL1, CANBTR0, CANBTR1, CANIDAC, CANIDAR0-7, and CANIDMR0-7 can only be written by the Power Architecture when the MSCAN is in initialization mode (INITRQ = 1 and INITAK = 1). The values of the error counters are not affected by initialization mode. When this bit is cleared by the Power Architecture, the MSCAN restarts and then tries to synchronize to the CAN bus. If the MSCAN is not in bus-off state, it synchronizes after 11 consecutive recessive bits on the CAN bus; if the MSCAN is in bus-off state, it continues to wait for 128 occurrences of 11 consecutive recessive bits. Writing to other bits in CANCTL0, CANRFLG, CANRIER, CANTFLG, or CANTIER must be done only after initialization mode is exited, which is INITRQ = 0 and INITAK = 0. 0 Normal operation. 1 MSCAN in initialization state.

1 The MSCAN must be in normal mode for this bit to become set.

2 See the Bosch CAN 2.0A/B specification for a detailed definition of transmitter and receiver states.

3 The Power Architecture has to make sure that the WUPE register and the WUPIE wake-up interrupt enable register (see Section 22.3.2.6, “MSCAN Receiver Interrupt Enable Register (CANRIER)”) is enabled, if the recovery mechanism from deep sleep mode is required.

4 The Power Architecture cannot clear SLPRQ before the MSCAN has entered sleep mode (SLPRQ = 1 and SLPAK = 1).

5 The Power Architecture cannot clear INITRQ before the MSCAN has entered initialization mode (INITRQ = 1 and INITAK = 1).

6 To protect from accidentally violating the CAN protocol, the TXCAN pin is immediately forced to a recessive state when the initialization mode is requested by the Power Architecture. Thus, the recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPAK = 1) before requesting initialization mode.

7 Not including WUPE, INITRQ, and SLPRQ.

8 TSTAT1 and TSTAT0 are not affected by initialization mode.

9 RSTAT1 and RSTAT0 are not affected by initialization mode.

22.3.2.2 MSCAN Control 1 Register (CANCTL1)

This register provides for various control and handshake status information of the MSCAN module as described below.

Read: Anytime

Write: The CLKSRC, LOOPB, LISTEN, BORM, and WUPE (in CANCTL0) bits can be written any time when in initialization mode (INITRQ = 1 and INITAK = 1).

Address: Base + 0x01 Access: User read/write

	0	1	2	3	4	5	6	7
R	CANE	CLKSRC	LOOPB	LISTEN	BORM	0	SLPAK	INITAK
W								
Reset	0	0	0	1	0	0	0	1

Figure 22-4. MSCAN Control 1 Register (CANCTL1)

Table 22-5. CANCTL1 field descriptions

Field	Description
CANE	MSCAN Enable 0 The MSCAN module is disabled. 1 The MSCAN module is enabled.
CLKSRC	MSCAN Clock Source. This bit defines the clock source for the MSCAN module (only for systems with a clock generation module; see Section 22.4.5, “Clock System,” and Figure 22-37). 0 The MSCAN clock source is the bus clock that can be originated from one of four sources: see Section 5.2.4, “MSCAN Clock Generation.” 1 The MSCAN clock source is the ips clock.
LOOPB	Loop Back Self Test Mode. When this bit is set, the MSCAN performs an internal loop-back that can be used for self test operation. The Tx bitstream output feeds back internally to the receiver. The RxCAN input pin is ignored and the TxCAN output goes to a recessive state (logic 1). The MSCAN behaves as it does normally when transmitting and treats its own transmitted message as a message received from a remote node. In this state, the MSCAN ignores bits sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both Tx and Rx interrupts are generated. 0 Loop Back Self Test disabled. 1 Loop Back Self Test enabled.
LISTEN	Listen Only Mode. This bit configures the MSCAN as a CAN bus monitor. When LISTEN is set, all valid CAN messages with matching IDs are received, but no acknowledgement or error frames are sent out (see Section 22.4.7.5, “Listen-Only Mode”). In addition, the error counters are frozen. Listen only mode supports applications that require hot plugging or throughput analysis. The MSCAN is unable to transmit any messages when listen only mode is active. 0 Normal operation. 1 Listen Only mode activated.
BORM	Bus-Off Recovery Mode. This bit configures the bus-off state recovery mode of the MSCAN. Refer to Section 22.4.15, “Bus-Off Recovery,” for details. 0 Automatic bus-off recovery. 1 Bus-Off recovery upon request.

Table 22-5. CANCTL1 field descriptions (Continued)

Field	Description
SLPAK	Sleep Mode Acknowledge. This flag indicates whether the MSCAN module has entered sleep mode (see Section 22.4.8.1, “MSCAN Sleep Mode”). It is used as a handshake flag for the SLPRQ sleep mode request. Sleep mode is active when SLPRQ = 1 and SLPAK = 1. Depending on the setting of WUPE, the MSCAN clears the flag if it detects activity on the CAN bus while in sleep mode. 0 Running.The MSCAN operates normally. 1 Sleep Mode Active.The MSCAN has entered sleep mode.
INITAK	Initialization Mode Acknowledge. This flag indicates whether the MSCAN module is in initialization mode (see Section 22.4.8.2, “MSCAN Initialization Mode”). It is used as a handshake flag for the INITRQ initialization mode request. Initialization mode is active when INITRQ = 1 and INITAK = 1. The registers CANCTL1, CANBTR0, CANBTR1, CANIDAC, CANIDAR0–CANIDAR7, and CANIDMR0–CANIDMR7 can be written only by the Power Architecture when the MSCAN is in initialization mode. 0 Running. The MSCAN operates normally. 1 Initialization Mode Active. The MSCAN has entered initialization mode.

22.3.2.3 MSCAN Bus Timing Register 0 (CANBTR0)

This register provides for various bus timing control of the MSCAN module as described below.

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

Address: Base + 0x04 Access: User read/write

	0	1	2	3	4	5	6	7
R W	SJW[1:0]		BRP[5:0]					
Reset	0	0	0	0	0	0	0	0

Figure 22-5. MSCAN Bus Timing Register 0 (CANBTR0)

Table 22-6. CANBTR0 field descriptions

Field	Description
SJW[1:0]	Synchronization Jump Width defines the maximum number of time quanta (Tq) clock cycles (refer to Figure 22-37 for Tq clock definition) a bit can be shortened or lengthened to achieve re-synchronization to data transitions on the bus. 00 1 Tq clock cycle. 01 2 Tq clock cycles. 10 3 Tq clock cycles. 11 4 Tq clock cycles.
BRP[5:0]	Baud Rate Prescaler bits determine time quanta (Tq) clock used to build up individual bit timing 000000 1 000001 2 000010 3 ... 111110 63 111111 64

22.3.2.4 MSCAN Bus Timing Register 1 (CANBTR1)

This register provides for various bus timing control of the MSCAN module as described below.

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

Address: Base + 0x05 Access: User read/write

	0	1	2	3	4	5	6	7
R W	SAMP	TSEG2[2:0]			TSEG1[3:0]			
Reset	0	0	0	0	0	0	0	0

Figure 22-6. MSCAN Bus Timing Register 1 (CANBTR1)

Table 22-7. CANBTR1 field descriptions

Field	Description
SAMP	Sampling. This bit determines the number of CAN bus samples taken per bit time. (see Figure 22-38). 0 One sample per bit. 1 Three samples per bit. If SAMP = 0, the resulting bit value is equal to the value of the single bit positioned at the sample point. If SAMP = 1, the resulting bit value is determined by using majority rule on the three total samples. For higher bit rates, it is recommended that only one sample is taken per bit time (SAMP = 0).
TSEG2[2:0]	Time Segment 2. Time segments within the bit-time fix the number of clock cycles per bit-time and location of the sample point (see Figure 22-38). 000 1 Tq clock cycle 001 2 Tq clock cycles 010 3 Tq clock cycles ..... 111 8 Tq clock cycles
TSEG1[3:0]	Time Segment 1. Time segments within the bit-time fix the number of clock cycles per bit-time and location of the sample point (see Figure 22-38). 0000 1 Tq clock cycle 0001 2 Tq clock cycles 0010 3 Tq clock cycles 0011 4 Tq clock cycles ..... 1110 15 Tq clock cycles 1111 16 Tq clock cycles

Bit time is determined by:

- Oscillator frequency
- Baud rate prescaler
- Number of time quanta (Tq) clock cycles per bit

Eqn. 22-1

$$\text{Bit Time} = \frac{(\text{Prescaler value})}{f_{CANCLK}} \times (1 + \text{TimeSegment1} + \text{Timesegment2})$$

22.3.2.5 MSCAN Receiver Flag Register (CANRFLG)

A flag can only be cleared when the condition that caused the setting is no longer valid and can only be cleared by software (writing a 1 to the corresponding bit position). Every flag has an associated interrupt enable bit in the CANRIER register.

The CANRFLG register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is left (INITRQ = 0 and INITAK = 0).

Read: Anytime

Write: Anytime when out of initialization mode, except RSTAT[1:0] and TSTAT[1:0] flags, which are read-only; write of 1 clears flag; write of 0 ignored

NOTE

The CANRFLG register is held in the reset state[1] when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

Address: Base + 0x08 Access: User read/write

	0	1	2	3	4	5	6	7
R	WUPIF	CSCIF	RSTAT[1:0]		TSTAT[1:0]		OVRIF	RXFIF
W	w1c	w1c					w1c	w1c
Reset	0	0	0	0	0	0	0	0

Figure 22-7. MSCAN Receiver Flag Register (CANRFLG)

Table 22-8. CANRFLG field descriptions

Field	Description
WUPIF	Wake-Up Interrupt Flag. If the MSCAN detects CAN bus activity while in sleep mode (see Section 22.4.8.1, “MSCAN Sleep Mode,”) and WUPE = 1 in CANTCTL0 (see Section 22.3.2.1, “MSCAN Control 0 Register (CANCTL0)”), the module sets WUPIF. If not masked, a wake-up interrupt is pending while this flag is set. 0 No wake-up activity observed while in Sleep Mode. 1 MSCAN detected activity on the bus and requested wake-up.
CSCIF	CAN Status Change Interrupt Flag. This flag is set when the MSCAN changes its current CAN bus status due to the actual value of the transmit error counter (TEC) and the receive error counter (REC). An additional 4-bit (RSTAT[1:0], TSTAT[1:0]) status register, which is split into separate sections for TEC/REC, informs the system on the actual CAN bus status (see Section 22.3.2.6, “MSCAN Receiver Interrupt Enable Register (CANRIER)”). If not masked, an error interrupt is pending while this flag is set. CSCIF provides a blocking interrupt. That guarantees that the receiver/transmitter status bits (RSTAT/TSTAT) are only updated when no CAN status change interrupt is pending. If the TECs/RECs change their current value after the CSCIF is asserted, which would cause an additional state change in the RSTAT/TSTAT bits, these bits keep their status until the current CSCIF interrupt is cleared again. 0 No change in CAN bus status occurred since last interrupt 1 MSCAN changed current CAN bus status

1. The RSTAT[1:0], TSTAT[1:0] bits are not affected by initialization mode.

Table 22-8. CANRFLG field descriptions (Continued)

Field	Description
RSTAT[1:0]	Receiver Status Bits. The values of the error counters control the actual CAN bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate receiver related CAN bus status of the MSCAN. The coding for the bits RSTAT1, RSTAT0 is: 00 RxOK: 0 ≤ receive error counter ≤ 96 01 RxWRN: 96 < receive error counter ≤ 127 10 RxERR: 127 < receive error counter 11 Bus-off[1]: transmit error counter > 255
TSTAT[1:0]	Transmitter Status Bits. The values of the error counters control the actual CAN bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate transmitter related CAN bus status of the MSCAN. The coding for the bits TSTAT1, TSTAT0 is: 00 TxOK: 0 ≤ transmit error counter ≤ 96 01 TxWRN: 96 < transmit error counter ≤ 127 10 TxERR: 127 < transmit error counter ≤ 255 11 Bus-Off: transmit error counter > 255
OVRIF	Overrun Interrupt Flag is set when a data overrun condition occurs. If not masked, an error interrupt is pending while this flag is set. 0 No data overrun condition. 1 A data overrun detected.
RXFIF	Receive Buffer Full Flag is set by MSCAN when a new message is shifted into the RX FIFO. Flag indicates whether the shifted buffer is loaded with a correctly received message (matching identifier, matching cyclic redundancy code (CRC) and no other errors detected). After Power Architecture reads the message from the RxFG buffer in the Rx FIFO, the RxF flag must be cleared to release the buffer. A set RxF flag prohibits shifting of the next FIFO entry into the foreground buffer (RxFG). If not masked, an RX interrupt is pending while this flag is set. To ensure data integrity, do not read the Rx buffer registers while RXFIF flag is cleared. 0 No new message available within the RxFG. 1 The receiver FIFO is not empty. A new message is available in the RxFG buffer.

1 Redundant Information for the most critical CAN bus status which is bus-off. This only occurs if the Tx error counter exceeds a number of 255 errors. Bus-off affects the receiver state. As soon as the transmitter leaves its bus-off state the receiver state skips to RxOK too. Refer also to TSTAT[1:0] coding in this register.

22.3.2.6 MSCAN Receiver Interrupt Enable Register (CANRIER)

This register contains the interrupt enable bits for the interrupt flags described in the CANRFLG register.

Read: Anytime

Write: Anytime when out of initialization mode

NOTE

WUPIE, CSCIE, OVRIE, and RXFIE are held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

Address: Base + 0x09 Access: User read/write

	0	1	2	3	4	5	6	7
R W	WUPIE	CSCIE	RSTATE[1:0]		TSTATE[1:0]		OVRIE	RXFIE
Reset	0	0	0	0	0	0	0	0

Figure 22-8. MSCAN Receiver Interrupt Enable Register (CANRIER)

Table 22-9. CANRIER field descriptions

Field	Description
WUPIE[1]	Wake-Up Interrupt Enable 0 No interrupt request is generated from this event. 1 A wake-up event causes a Wake-Up interrupt request.
CSCIE	CAN Status Change Interrupt Enable 0 No interrupt request is generated from this event. 1 A CAN Status Change event causes an error interrupt request.
RSTATE[1:0]	Receiver Status Change Enable. These RSTAT enable bits control the sensitivity level in which receiver state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level the RSTAT flags continue to indicate the actual receiver state and are only updated if no CSCIF interrupt is pending. 00 Do not generate any CSCIF interrupt caused by receiver state changes. 01 Generate CSCIF interrupt only if the receiver enters or leaves Bus-Off[2] state. Discard other receiver state changes for generating CSCIF interrupt. 10 Generate CSCIF interrupt only if the receiver enters or leaves RxErr or Bus-Off state. Discard other receiver state changes for generating CSCIF interrupt. 11 Generate CSCIF interrupt on all state changes
TSTATE[1:0]	Transmitter Status Change Enable. These TSTAT enable bits control the sensitivity level in which transmitter state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level, the TSTAT flags continue to indicate the actual transmitter state and are only updated if no CSCIF interrupt is pending. 00 Do not generate any CSCIF interrupt caused by transmitter state changes. 00 Do not generate any CSCIF interrupt caused by transmitter state changes. 01 Generate CSCIF interrupt only if the transmitter enters or leaves Bus-Off state. Discard other transmitter state changes for generating CSCIF interrupt. 10 Generate CSCIF interrupt only if the transmitter enters or leaves TxErr or Bus-Off state. Discard other transmitter state changes for generating CSCIF interrupt. 11 Generate CSCIF interrupt on all state changes
OVRIE	Overrun Interrupt Enable 0 No interrupt request is generated from this event. 1 An overrun event causes an error interrupt request.
RXFIE	Receive Buffer Full Interrupt Enable 0 No interrupt request is generated from this event. 1 A receive buffer full (successful message reception) event causes a receiver interrupt request.

1 WUPIE and WUPE (see Section 22.3.2.1, “MSCAN Control 0 Register (CANCTL0)”) must both be enabled if the recovery mechanism from deep sleep mode is required.

2 Bus-off state is defined by the CAN standard (see Bosch CAN 2.0A/B protocol specification: for only transmitters. Because the only possible state change for the transmitter from bus-off to TxOK also forces the receiver to skip its current state to RxOK, the coding of the RXSTAT[1:0] flags define an additional bus-off state for the receiver (see Section 22.3.2.5, “MSCAN Receiver Flag Register (CANRFLG)”).

22.3.2.7 MSCAN Transmitter Flag Register (CANTFLG)

The transmit buffer empty flags each have an associated interrupt enable bit in the CANTIER register.

Read: Anytime

Write: Anytime for TXEx flags when not in initialization mode; write of 1 clears flag, write of 0 ignored

NOTE

The CANTFLG register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is left (INITRQ = 0 and INITAK = 0).

Address: Base + 0x0C Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	TXE2IF	TXE1IF	TXE0IF
W						w1c	w1c	w1c
Reset	0	0	0	0	0	1	1	1

Figure 22-9. MSCAN Transmitter Flag Register (CANTFLG)

Table 22-10. CANTFLG field descriptions

Field	Description
TXE*x*IF	Transmitter Buffer Empty. These flags indicate that the associated transmit message buffer is empty, and thus not scheduled for transmission. The Power Architecture must clear the flag after a message is set up in the transmit buffer and is due for transmission. The MSCAN sets the flag after the message is sent successfully. The flag is also set by the MSCAN when the transmission request is successfully aborted due to a pending abort request (see Section 22.3.2.9, “MSCAN Transmitter Message Abort Request (CANTARQ)”). If not masked, a transmit interrupt is pending while this flag is set. Clearing a TXE*x*IF flag also clears the corresponding ABTAK*x* (see Section 22.3.2.10, “MSCAN Transmitter Message Abort Acknowledge (CANTAAK)”). When a TXE*x*IF flag is set, the corresponding ABTRQ*x* bit is cleared (see Section 22.3.2.9, “MSCAN Transmitter Message Abort Request (CANTARQ)”). When listen-mode is active (see Section 22.3.2.2, “MSCAN Control 1 Register (CANCTL1)”) the TXE*x*IF flags cannot be cleared and no transmission is started. Read and write accesses to the transmit buffer are blocked if the corresponding TXE*x*IF bit is cleared (TXE*x*IF = 0) and the buffer is scheduled for transmission. 0 The associated message buffer is full (loaded with a message due for transmission). 1 The associated message buffer is empty (not scheduled).

22.3.2.8 MSCAN Transmitter Interrupt Enable Register (CANTIER)

This register contains the interrupt enable bits for the transmit buffer empty interrupt flags.

Read: Anytime

Write: Anytime when not in initialization mode

NOTE

The CANTIER register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is left (INITRQ = 0 and INITAK = 0).

Address: Base + 0x0D Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	TXEIE2	TXEIE1	TXEIE0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-10. MSCAN Transmitter Interrupt Enable Register (CANTIER)

Table 22-11. CANTIER field descriptions

Field	Description
TXEIE[2:0]	Transmitter Empty Interrupt Enable. 0 No interrupt request is generated from this event. 1 A transmitter empty (transmit buffer available for transmission) event causes a transmitter empty interrupt request.

22.3.2.9 MSCAN Transmitter Message Abort Request (CANTARQ)

This register provides for abort request of queued messages as described below.

Read: Anytime

Write: Anytime when not in initialization mode

NOTE

The CANTARQ register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is left (INITRQ = 0 and INITAK = 0).

Address: Base + 0x10 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-11. MSCAN Transmitter Interrupt Enable Register (CANTIER)

Table 22-12. CANTIER field descriptions

Field	Description
ABTRQ[2:0]	Abort Request. The Power Architecture sets the ABTRQ*x* bit to request that a scheduled message buffer (TXExIF = 0) be aborted. The MSCAN grants the request if the message has not already started transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted, the associated TXExIF (see Section 22.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and abort acknowledge flags (ABTAK, see Section 22.3.2.10, “MSCAN Transmitter Message Abort Acknowledge (CANTAAK)”) are set and a transmit interrupt occurs, if enabled. The Power Architecture cannot reset ABTRQ*x*. ABTRQ*x* is reset whenever the associated TXE*x*IF flag is set. 0 No abort request. 1 Abort request pending.

22.3.2.10 MSCAN Transmitter Message Abort Acknowledge (CANTAAK)

The CANTAAK register indicates the successful abort of a queued message if requested by the appropriate bits in the CANTARQ register.

Read: Anytime

Write: Unimplemented for ABTAKx flags

NOTE

The CANTAAK register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1).

Address: Base + 0x11 Access: User read-only

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	ABTAK2	ABTAK1	ABTAK0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-12. MSCAN Transmitter Message Abort Acknowledge (CANTAAK)

Table 22-13. CANTAAK field descriptions

Field	Description
ABTAK[2:0]	Abort Acknowledge flag acknowledges that a message was aborted due to a pending Power Architectureabort request. After a specific message buffer is flagged empty, application software can use this flag to identify whether the message was successfully aborted or was sent. This flag is cleared when the corresponding TxE flag is cleared. 0 The message was not aborted. 1 The message was aborted.

22.3.2.11 MSCAN Transmitter Buffer Selection (CANTBSEL)

This register allows the selection of the actual transmit message buffer, which is then accessible in the CANTXFG register space.

Read: Find the lowest ordered bit set to 1, all other bits are read as 0

Write: Anytime when not in initialization mode

NOTE

The CANTBSEL register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again upon exiting the initialization mode (INITRQ = 0 and INITAK = 0).

The following gives a short programming example of usage of the CANTBSEL register:

To get the next available transmit buffer, application software must read the CANTFLG register and write this value back into the CANTBSEL register. In this example Tx buffers TX1 and TX2 are available. The value read from CANTFLG is therefore 0b0000_0110. When writing this value back to CANTBSEL, the Tx buffer TX1 is selected in the CANTXFG because the lowest numbered bit set to 1 is at bit position 1. Reading back this value out of CANTBSEL results in 0b0000_0010, because only the lowest numbered bit position set to 1 is presented. This mechanism eases the application software the selection of the next available Tx buffer.

- LDD CANTFLG; value read is 0b0000_0110
- STD CANTBSEL; value written is 0b0000_0110
- LDD CANTBSEL; value read is 0b0000_0010

If all transmit message buffers are deselected, no accesses are allowed to the CANTXFG registers.

Address: Base + 0x14 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	TX2	TX1	TX0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-13. MSCAN Transmitter Buffer Selection (CANTBSEL)

Table 22-14. CANTBSEL field descriptions

Field	Description
TX[2:0]	Transmit Buffer Select. The lowest numbered bit places the respective transmit buffer in the CANTXFG register space (e.g., TX1 = 1 and TX0 = 1 selects transmit buffer TX0; TX1 = 1 and TX0 = 0 selects transmit buffer TX1). Read and write accesses to the selected transmit buffer are blocked if the corresponding TXEx bit is cleared and the buffer is scheduled for transmission (see Section 22.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”). 0 The associated message buffer is deselected. 1 The associated message Buffer is selected, if lowest numbered bit.

22.3.2.12 MSCAN Identifier Acceptance Control Register (CANIDAC)

This register provides for identifier acceptance control as described below.

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1), except bits IDHIT*x* which are read-only

The IDHIT*x* indicators are always related to the message in the foreground buffer (RxFG). When a message gets shifted into the foreground buffer of the receiver FIFO, the indicators are updated as well.

Address: Base + 0x15 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	IDAM[1:0]		0	IDHIT[2:0]		
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-14. MSCAN Identifier Acceptance Control Register (CANIDAC)

Table 22-15. CANIDAC field descriptions

Field	Description
IDAM[1:0]	Identifier Acceptance Mode. Power Architecture sets these flags to define the identifier acceptance filter organization (see Section 22.4.3, “Identifier Acceptance Filter”). In filter closed mode, no message is accepted so the foreground buffer is never reloaded. 00 Two 32-bit acceptance filters. 01 Four 16-bit acceptance filters. 10 Eight 8-bit acceptance filters. 11 Filter closed.
IDHIT[2:0]	Identifier Acceptance Hit Indicator. MSCAN sets these flags to indicate an identifier acceptance hit (see Section 22.4.3, “Identifier Acceptance Filter”). 000 Filter 0 hit. 001 Filter1 hit. 010 Filter 2 hit. 011 Filter 3 hit. 100 Filter 4 hit. 101 Filter 5 hit. 110 Filter 6 hit. 111 Filter 7 hit.

22.3.2.13 MSCAN MISC Register (CANMISC)

Read: Anytime

Write: Anytime; write of 1 clears flag; write of 0 ignored.

Address: Base + 0x19 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	BOFFHOLD
W								w1c
Reset	0	0	0	0	0	0	0	0

Figure 22-15. MSCAN MISC Register (CANMISC)

Table 22-16. CANMISC field descriptions

Field	Description
BOFFHOLD	Bus-off State Hold Until User Request. If BORM is set in MSCAN Control Register 1 (see Section 22.3.2.2, "MSCAN Control 1 Register (CANCTL1)"), this bit indicates whether the module has entered the bus-off state. Clearing this bit requests the recovery from bus-off. Refer to Section 22.4.15, "Bus-Off Recovery," for details. 0 Module is not bus-off or recovery has been requested by user in bus-off state. 1 Module is bus-off and holds this state until user request.

22.3.2.14 MSCAN Receive Error Counter Register (CANRXERR)

This register reflects the status of the MSCAN receive error counter.

Read: Only when in sleep mode (SLPRQ = 1 and SLPAK = 1) or initialization mode (INITRQ = 1 and INITAK = 1).

Write: Unimplemented

NOTE

Reading this register when in any other mode other than sleep or initialization mode may return an incorrect value.

Address: Base + 0x1C Access: User read-only

	0	1	2	3	4	5	6	7
R	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-16. MSCAN Receive Error Counter Register (CANRXERR)

Table 22-17. CANRXERR field descriptions

Field	Description
RXERR[7:0]	This register reflects the status of the MSCAN receive error counter. Read: Only when in sleep mode (SLPRQ = 1 and SLPAK = 1) or initialization mode (INITRQ = 1 and INITAK = 1)

22.3.2.15 MSCAN Transmit Error Counter Register (CANTXERR)

This register reflects the status of the MSCAN transmit error counter.

Read: Only when in sleep mode (SLPRQ = 1 and SLPAK = 1) or initialization mode (INITRQ = 1 and INITAK = 1)

Write: Unimplemented

NOTE

Reading this register when in any other mode other than sleep or initialization mode, may return an incorrect value.

Address: Base + 0x1D Access: User read-only

	0	1	2	3	4	5	6	7
R	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-17. MSCAN Transmit Error Counter Register (CANTXERR)

Table 22-18. CANTXERR field descriptions

Field	Description
TXERR[7:0]	This register reflects the status of the MSCAN receive error counter. Read: Only when in sleep mode (SLPRQ = 1 and SLPAK = 1) or initialization mode (INITRQ = 1 and INITAK = 1)

22.3.2.16 MSCAN Identifier Acceptance Register (CANIDAR0–CANIDAR7)

On reception, each message is written into the background receive buffer. The Power Architecture is signaled to read the message only if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message is overwritten by the next message (dropped).

Address: Base + 0x20 (CANIDAR0)
Base + 0x21 (CANIDAR1)
Base + 0x24 (CANIDAR2)
Base + 0x25 (CANIDAR3)
Base + 0x30 (CANIDAR4)
Base + 0x31 (CANIDAR5)
Base + 0x34 (CANIDAR6)
Base + 0x35 (CANIDAR7)
Access: User read/write

	0	1	2	3	4	5	6	7
R W	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Reset	0	0	0	0	0	0	0	0

Figure 22-18. MSCAN Identifier Acceptance Registers (CANIDAR0–CANIDAR7)

Table 22-19. CANIDAR0–CANIDAR7 field descriptions

Field	Description
AC[7:0]	Acceptance Code Bits. AC[7:0] comprises a user defined sequence of bits with which the corresponding bits of the related identifier register (IDR*n*) of the receive message buffer are compared. The result of this comparison is masked with the corresponding identifier mask register.

The acceptance registers of the MSCAN are applied on the IDR0–IDR3 registers (see Section 22.3.3.1, “Identifier Registers (IDR0–IDR3)”) of incoming messages in a bit-by-bit manner (see Section 22.4.3, “Identifier Acceptance Filter”).

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers, only the first two (CANIDAR0/1, CANIDMR0/1) are applied.

22.3.2.17 MSCAN Identifier Mask Register (CANIDMR0–CANIDMR7)

Address: Base + 0x28 (CANIDMR0)
Base + 0x29 (CANIDMR1)
Base + 0x24 (CANIDMRC)
Base + 0x25 (CANIDMRD)
Base + 0x38 (CANIDMR4)
Base + 0x39 (CANIDMR5)
Base + 0x3C (CANIDMR6)
Base + 0x3D (CANIDMR7)
Access: User read/write

	0	1	2	3	4	5	6	7
R W	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset	0	0	0	0	0	0	0	0

Figure 22-19. MSCAN Identifier Mask Registers (CANIDMR0–CANIDMR7)

Table 22-20. CANIDMR0–CANIDMR7 field descriptions

Field	Description
AM[7:0]	Acceptance Mask Bits. If a particular bit in this register is cleared, this indicates the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates the state of the corresponding bit in the identifier acceptance register does not affect whether or not message is accepted. 0 Match corresponding acceptance code register and identifier bits. 1 Ignore corresponding acceptance code register bit.

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering.

- To receive standard identifiers in 32-bit filter mode, the last three bits (AM[0:2]) in the following mask registers must be programmed as “don’t care:”
  - CANIDMR1
  - CANIDMR5
- To receive standard identifiers in 16-bit filter mode, the last three bits (AM[0:2]) in the following mask registers must be programmed as “don’t care:”
  - CANIDMR1
  - CANIDMR3
  - CANIDMR5
  - CANIDMR7

22.3.3 Programmer’s Model of Message Storage

The following section details the organization of the receive and transmit message buffers and the associated control registers.

Receive buffer start with base address 0x40, Transmit buffer start with base address 0x60.

To simplify the programmer interface, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 bytes in the memory map containing a 13-byte data structure.

An additional transmit buffer priority register (TBPR) is defined for the transmit buffers. Within the last two bytes of this memory map, the MSCAN stores a special 16-bit time stamp, which is sampled from an internal timer after successful transmission or reception of a message. This feature is only available for transmit and receiver buffers, if the TIME bit is set (see Section 22.3.2.1, “MSCAN Control 0 Register (CANCTL0)”).

The time stamp register is written by the MSCAN. The Power Architecture can only read these registers.

Table 22-21. Message Buffer Organization

Offset Address Base + 0x40 + offset (RX) Base + 0x60 + offset (TX)	Register
0x00	Identifier Register 0
0x01	Identifier Register 1
0x04	Identifier Register 2
0x05	Identifier Register 3
0x08	Data Segment Register 0
0x09	Data Segment Register 1
0x0C	Data Segment Register 2
0x0D	Data Segment Register 3
0x10	Data Segment Register 4
0x11	Data Segment Register 5
0x14	Data Segment Register 6
0x15	Data Segment Register 7
0x18	Data Length Register
0x19	Transmit Buffer Priority Register[1]
0x1C	Time Stamp Register (High Byte)[2]
0x1D	Time Stamp Register (Low Byte)[2]

1 Not applicable for receive buffers.

2 Read-only.

Figure 22-20 shows the common 13-byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in Figure 22-21.

All bits of the receive and transmit buffers are of indeterminate value out of reset because of RAM-based implementation[1]. All reserved or unused bits of the receive and transmit buffers always read an indeterminate value.

1. Exception: The transmit priority registers are 0 out of reset.

Address: Base + 0x40 (RX)
Base + 0x60 (TX)

		0	1	2	3	4	5	6	7
0x00 IDR0	R	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
	W								
0x01 IDR1	R	ID20	ID19	ID18	SRR (=1)	IDE (=1)	ID17	ID16	ID15
	W								
0x04 IDR2	R	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
	W								
0x05 IDR3	R	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
	W								
0x08 DSR0	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x09 DSR1	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x0C DSR2	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x0D DSR3	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x10 DSR4	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x11 DSR5	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x14 DSR6	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x15 DSR7	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x18 DLR	R					DLC3	DLC2	DLC1	DLC0
	W								

= Unused, always read ‘x’

Figure 22-20. Receive/Transmit Message Buffer—Extended Identifier Mapping

Read: For transmit buffers, anytime when TXEx flag is set (see Section 22.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 22.3.2.11, “MSCAN Transmitter Buffer Selection (CANTBSEL)”). For receive buffers, only when RXF flag is set (see Section 22.3.2.5, “MSCAN Receiver Flag Register (CANRFLG)”).

Write: For transmit buffers, anytime when TXEx flag is set (see Section 22.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 22.3.2.11, “MSCAN Transmitter Buffer Selection (CANTBSEL)”). Unimplemented for receive buffers.

Reset: Undefined because of RAM-based implementation.

Address: Base + 0x40 (RX)
Base + 0x60 (TX)

		0	1	2	3	4	5	6	7
IDR0 0x00	R	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
	W								
IDR1 0x01	R	ID2	ID1	ID0	RTR	IDE (=0)			
	W								

Figure 22-21. Receive/Transmit Message Buffer — Standard Identifier Mapping

22.3.3.1 Identifier Registers (IDR0–IDR3)

The identifier registers for an extended format identifier consist of a total of 32 bits; ID[28:0], SRR, IDE, and RTR bits. The identifier registers for a standard format identifier consist of a total of 13 bits; ID[10:0], RTR, and IDE bits.

22.3.3.1.1 IDR0–IDR3 for Extended Identifier Mapping

Address: Base + 0x40 (RX)
Base + 0x60 (TX)

Access: User read/write

	0	1	2	3	4	5	6	7
R	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
W								
Reset[1]	—	—	—	—	—	—	—	—

Figure 22-22. Identifier Register 0 (IDR0)— Extended Identifier Mapping

1 Reset value is indeterminate.

Table 22-22. IDR0 Field Descriptions—Extended Mapping

Field	Description
ID[28:21]	**Extended Format Identifier.** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

Address: Base + 0x41 (RX)
Base + 0x61 (TX)

Access: User read/write

	0	1	2	3	4	5	6	7
R W	ID20	ID19	ID18	SRR	IDE	ID17	ID16	ID15
Reset[1]	U	U	U	U	U	U	U	U

Figure 22-23. Identifier Register 1 (IDR1)—Extended Identifier Mapping

1 Reset value is indeterminate.

Table 22-23. IDR1 Register Field Descriptions—Extended Mapping

Field	Description
ID[20:18]	Extended Format Identifier. The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.
SRR	Substitute Remote Request. This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and is stored as received on the CAN bus for receive buffers.
IDE	ID Extended. This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the Power Architecture how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send. 0 Standard format (11 bit) 1 Extended format (29 bit)
ID[17:15]	Extended Format Identifier. The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

Address: Base + 0x44 (RX)
Base + 0x64 (TX)

Access: User read/write

	0	1	2	3	4	5	6	7
R W	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
Reset[1]	U	U	U	U	U	U	U	U

Figure 22-24. Identifier Register 2 (IDR2)—Extended Identifier Mapping

1 Reset value is indeterminate.

Table 22-24. IDR2 Register Field Descriptions — Extended Mapping

Field	Description
ID[14:7]	Extended Format Identifier. The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

Address: Base + 0x45 (RX)
Base + 0x65 (TX)

Access: User read/write

	0	1	2	3	4	5	6	7
R W	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
Reset[1]	U	U	U	U	U	U	U	U

Figure 22-25. Identifier Register 3 (IDR3)—Extended Identifier Mapping

1 Reset value is indeterminate.

Table 22-25. IDR3 Register Field Descriptions—Extended Mapping

Field	Description
ID[6:0]	Extended Format Identifier. The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.
RTR	Remote Transmission Request. This flag reflects the status of the remote transmission request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this flag defines the setting of the RTR bit to be sent. 0 Data frame 1 Remote frame

22.3.3.1.2 IDR0–IDR1 for Standard Identifier Mapping

Figure 22-26 and Figure 22-27 show the Identifier Registers for standard identifier mapping.

NOTE

With standard identifier mapping, registers IDR2 and IDR3 are not used.

Address: Base + 0x40 (RX)
Base + 0x60 (TX)

Access: User read/write

	0	1	2	3	4	5	6	7
R W	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
Reset[1]	U	U	U	U	U	U	U	U

Figure 22-26. Identifier Register 0 (IDR0)—Standard Mapping

1 Reset value is indeterminate.

Table 22-26. IDR0 Register Field Descriptions—Standard Mapping

Field	Description
ID[10:3]	Standard Format Identifier. The identifiers consist of 11 bits (ID[10:0]) for the standard format. ID10 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. See also ID bits in Table 22-27.

Address: Base + 0x41 (RX)
Base + 0x61 (TX)

Access: User read/write

	0	1	2	3	4	5	6	7
R W	ID2	ID1	ID0	RTR	IDE (=0)			
Reset[1]	—	—	—	—	—	—	—	—

Figure 22-27. Identifier Register 1 (IDR1)—Standard Mapping

1 Reset value is indeterminate.

Table 22-27. IDR1 Register Field Descriptions—Standard Mapping

Field	Description
ID[2:0]	Standard Format Identifier. The identifiers consist of 11 bits (ID[10:0]) for the standard format. ID10 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. See also ID bits in Table 22-26.
RTR	Remote Transmission Request. This flag reflects the status of the Remote Transmission Request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this flag defines the setting of the RTR bit to be sent. 0 Data frame. 1 Remote frame.
IDE	ID Extended. This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the Power Architecture how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send. 0 Standard format (11 bit). 1 Extended format (29 bit).

22.3.3.2 Data Segment Registers (DSR0–7)

The eight data segment registers, each with bits DB[7:0], contain the data to be transmitted or received. The number of bytes to be transmitted or received is determined by the data length code in the corresponding DLR register.

Address: 0x48 (RX DSR0) 0x68 (TX DSR0)
0x49 (RX DSR1) 0x69 (TX DSR1)
0x4C (RX DSR2) 0x6C (TX DSR2)
0x4D (RX DSR3) 0x6D (TX DSR3)
0x50 (RX DSR4) 0x70 (TX DSR4)
0x51 (RX DSR5) 0x71 (TX DSR5)
0x54 (RX DSR6) 0x74 (TX DSR6)
0x55 (RX DSR7) 0x75 (TX DSR7)

Access: User read/write

	0	1	2	3	4	5	6	7
R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Reset[1]	—	—	—	—	—	—	—	—

Figure 22-28. Data Segment Registers (DSR0–DSR7)—Extended Identifier Mapping

1 Reset value is indeterminate.

Table 22-28. DSR0–DSR7 field descriptions

Field	Description
DB[7:0]	Data bits 7:0

22.3.3.3 Data Length Register (DLR)

The Data Length Register (DLR) keeps the data length field of the CAN frame.

Address: Base + 0x58 (RX)
Base + 0x78 (TX)

Access: User read/write

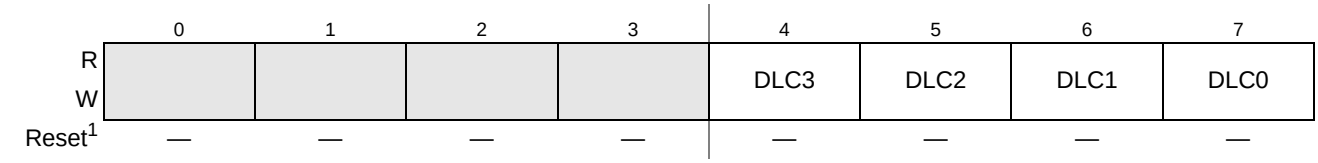


Figure 22-29. Data Length Register (DLR)—Extended Identifier Mapping

1 Reset value is indeterminate.

Table 22-29. DLR field descriptions

Field	Description
3:0 DLC[3:0]	Data Length Code Bits. The data length code contains the number of bytes (data byte count) of the respective message. During the transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted data bytes is always 0. The data byte count ranges from 0 to 8 for a data frame, as follows: 0000 0 data bytes. 0001 1 data byte. 0010 2 data bytes. 0011 3 data bytes. 0100 4 data bytes. 0101 5 data bytes. 0110 6 data bytes. 0111 7 data bytes. 1000 8 data bytes. All other values are reserved.

22.3.3.4 Transmit Buffer Priority Register (TBPR)

This register defines the local priority of the associated message buffer. The local priority is used for the internal prioritization process of the MSCAN and is defined to be highest for the smallest binary number. The MSCAN implements the following internal prioritization mechanisms:

- All transmission buffers with a cleared TXEx flag participate in the prioritization immediately before the SOF (start of frame) is sent.
- The transmission buffer with the lowest local priority field wins the prioritization.

In cases of more than one buffer having the same lowest priority, the message buffer with the lower index number wins.

Address: Base + 0x59 (RX)
Base + 0x79 (TX)

Access: User read/write

	0	1	2	3	4	5	6	7
R W	PRIO7	PRIO6	PRIO5	PRIO4	PRIO3	PRIO2	PRIO1	PRIO0
Reset	0	0	0	0	0	0	0	0

Figure 22-30. Transmit Buffer Priority Register (TBPR)

Read: Anytime when TXEx flag is set (see Section 22.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 22.3.2.11, “MSCAN Transmitter Buffer Selection (CANTBSEL)”).

Write: Anytime when TXEx flag is set (see Section 22.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 22.3.2.11, “MSCAN Transmitter Buffer Selection (CANTBSEL)”).

22.3.3.5 Time Stamp Register (TSRH–TSRL)

If the TIME bit is enabled, the MSCAN writes a special time stamp to the respective registers in the active transmit or receive buffer as soon as a message has been acknowledged on the CAN bus (see Section 22.3.2.1, “MSCAN Control 0 Register (CANCTL0)”). The time stamp is written on the bit sample point for the recessive bit of the ACK delimiter in the CAN frame. In case of a transmission, the Power Architecture can only read the time stamp after the respective transmit buffer has been flagged empty.

The timer value, which is used for stamping, is taken from a free running internal CAN bit clock. A timer overrun is not indicated by the MSCAN. The timer is reset (all bits set to 0) during initialization mode. The Power Architecture can only read the time stamp registers.

Address: Base + 0x5C (RX)
Base + 0x7C (TX)

Access: User read-only

	0	1	2	3	4	5	6	7
R	TSR15	TSR14	TSR13	TSR12	TSR11	TSR10	TSR9	TSR8
W								
Reset[1]	—	—	—	—	—	—	—	—

Figure 22-31. Time Stamp Register—High Byte (TSRH)

1 Reset value is indeterminate.

Address: Base + 0x5D (RX)
Base + 0x7D (TX)

Access: User read-only

	0	1	2	3	4	5	6	7
R	TSR7	TSR6	TSR5	TSR4	TSR3	TSR2	TSR1	TSR0
W								
Reset[1]	—	—	—	—	—	—	—	—

Figure 22-32. Time Stamp Register— Low Byte (TSRL)

[1] Reset value is indeterminate.

Table 22-30. TSRH–TSRL field descriptions

Field	Description
TSR[15:0]	Time stamp data bits 15:0

Read: Anytime when TXEx flag is set (see Section 22.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 22.3.2.11, “MSCAN Transmitter Buffer Selection (CANTBSEL)”).

Write: Unimplemented

22.4 Functional Description

22.4.1 General

This section provides a complete functional description of the MSCAN.

22.4.2 Message Storage

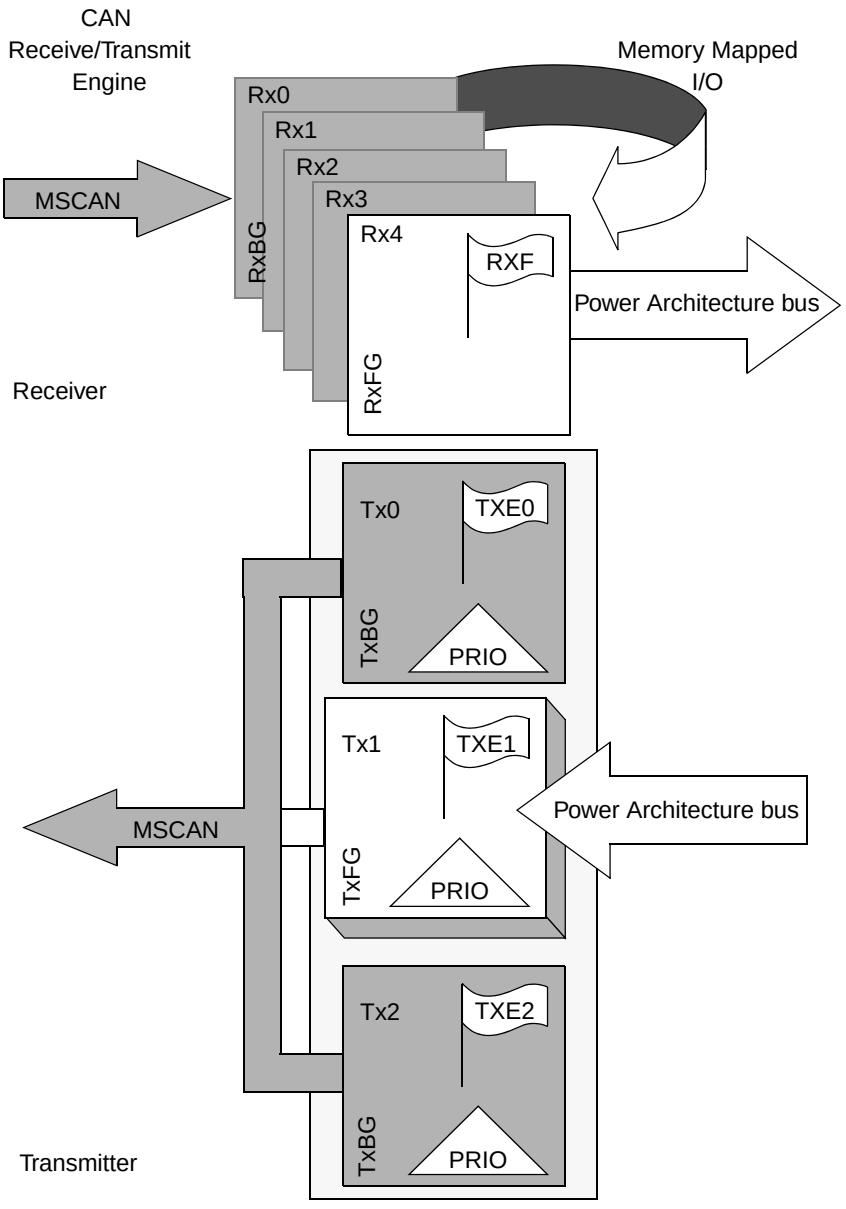


Figure 22-33. User Model for Message Buffer Organization

MSCAN facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

22.4.2.1 Message Transmit Background

Modern application layer software is built upon two fundamental assumptions:

- Any CAN node is able to send out a stream of scheduled messages without releasing the bus between the two messages. Such nodes arbitrate for the bus immediately after sending the previous message and only release the bus in case of lost arbitration.
- The internal message queue within any CAN node is organized so the highest priority message is sent out first if more than one message is ready to be sent.

The above behavior cannot be achieved with a single transmit buffer. That buffer must be reloaded right after the previous message is sent. This loading process lasts a finite amount of time and has to be completed within the inter-frame sequence (IFS)[1] to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires the Power Architecture to react with short latencies to the transmit interrupt.

A double buffer scheme decouples the reloading of the transmit buffer from the actual message sending and reduces the reactiveness requirements on the Power Architecture. Problems can arise if the sending of a message is finished while the Power Architecture reloads the second buffer. No buffer would be ready for transmission and the bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the MSCAN implements with the local priority concept described in Section 22.4.2.2, “Transmit Structures.”

22.4.2.2 Transmit Structures

The MSCAN has a triple transmit buffer scheme which allows multiple messages to be set up in advance and achieve an optimized real-time performance. The three buffers are arranged as shown in Figure 22-33.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see Section 22.3.3, “Programmer’s Model of Message Storage”). An additional Section 22.3.3.4, “Transmit Buffer Priority Register (TBPR),” contains an 8-bit local priority field (PRIO). The remaining two bytes are used for time stamping of a message, if required (see Section 22.3.3.5, “Time Stamp Register (TSRH–TSRL)”).

To transmit a message, the Power Architecture must identify an available transmit buffer, which is indicated by a set transmitter buffer empty (TXEx) flag (see Section 22.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”). If a transmit buffer is available, the Power Architecture must set a pointer to this buffer by writing to the CANTBSEL register (see Section 22.3.2.11, “MSCAN Transmitter Buffer Selection (CANTBSEL)”). This makes the respective buffer accessible within the CANTXFG address space (see Section 22.3.3, “Programmer’s Model of Message Storage”). The algorithmic feature associated with the CANTBSEL register simplifies the transmit buffer selection. In addition, this scheme makes the handler software simpler because only one address area is applicable for the transmit process, and the required address space is minimized.

1. Reference the Bosch CAN 2.0A/B protocol specification dated September 1991.

The Power Architecture then stores the identifier, the control bits and the data content into one of the transmit buffers. Finally, the buffer is flagged as ready for transmission by clearing the associated TXE flag.

The MSCAN then schedules the message for transmission and signals the successful transmission of the buffer by setting the associated TXE flag. A transmit interrupt (see Section 22.4.11.1, “Transmit Interrupt”) is generated[1] when TXEx is set and can drive the application software to reload the buffer.

In case more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN uses the local priority setting of the three buffers to determine the prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software programs this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages transmitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority. The internal scheduling process takes place when the MSCAN arbitrates for the bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message in one of the three transmit buffers. Because messages that are already in transmission cannot be aborted, the user must request the abort by setting the corresponding abort request bit (ABTRQ) (see Section 22.3.2.9, “MSCAN Transmitter Message Abort Request (CANTARQ)”). The MSCAN then grants the request, if possible, by:

1. Setting the corresponding abort acknowledge flag (ABTAK) in the CANTAAK register.
2. Setting the associated TXE flag to release the buffer.
3. Generating a transmit interrupt. The transmit interrupt handler software can determine from the setting of the ABTAK flag whether the message was aborted (ABTAK = 1) or sent (ABTAK = 0).

22.4.2.3 Receive Structures

The received messages are stored in a five-stage input FIFO. The five message buffers are alternately mapped into a single memory area as shown in Figure 22-33. While the background receive buffer (RxBG) is exclusively associated with the MSCAN, the foreground receive buffer (RxFG) is addressable by the Power Architecture as seen in Figure 22-33. This scheme simplifies the manager software as only one address area is applicable for the receive process.

All receive buffers have a size of 15 bytes to store the CAN control bits, the identifier (standard or extended), the data contents, and a time stamp, if enabled (for details, see Section 22.4.3, “Identifier Acceptance Filter”)[2].

The receiver full flag (RXFIF) described in Section 22.3.2.5, “MSCAN Receiver Flag Register (CANRFLG),” signals the status of the foreground receive buffer. When the buffer contains a correctly received message with a matching identifier, this flag is set.

On reception, each message is checked to see if it passes the filter (see Section 22.3.2.12, “MSCAN Identifier Acceptance Control Register (CANIDAC)”) and in parallel, is written into the active RxBG. After successful reception of a valid message, the MSCAN shifts the content of RxBG into the receiver

1. The transmit interrupt occurs only if not masked. A polling scheme can be applied on TXEx also.
2. Reference the Bosch CAN 2.0A/B protocol specification dated September 1991 for details.

FIFO[1], and generates a receive interrupt (see Section 22.4.11.2, “Receive Interrupt”) to the Power Architecture[2] by set RXFIF. The receive manager has to read the received message from the RxFG, reset the RXFIF flag to acknowledge the interrupt, and release the foreground buffer. A new message, which can follow immediately after the IFS field of the CAN frame, is received into the next available RxBG. If the MSCAN receives an invalid message in its RxBG (wrong identifier, transmission errors etc.), the actual contents of the buffer are over-written by the next message. The buffer is not shifted into the FIFO.

When the MSCAN module is transmitting, the MSCAN receives its own transmitted messages into the background receive buffer (RxBG), but does not shift it into the receiver FIFO, generate a receive interrupt, or acknowledge its own messages on the CAN bus. The exception to this rule is in loop back mode Section 22.3.2.2, “MSCAN Control 1 Register (CANCTL1),” where the MSCAN treats its own messages exactly like all other incoming messages. The MSCAN receives its own transmitted messages in the event that it loses arbitration[3]. If arbitration is lost, the MSCAN must be prepared to become a receiver.

An overrun condition occurs when all receive message buffers in the FIFO are filled with correctly received messages with accepted identifiers and another message is correctly received from the bus with an accepted identifier. The latter message is discarded and an error interrupt with overrun indication is generated if enabled Section 22.4.11.4, “Error Interrupt.” The MSCAN remains able to transmit messages while the receiver FIFO is being filled, but all incoming messages are discarded. As soon as a receive buffer in the FIFO is available again, new valid messages are accepted.

22.4.3 Identifier Acceptance Filter

The MSCAN identifier acceptance registers (Section 22.3.2.12, “MSCAN Identifier Acceptance Control Register (CANIDAC)”) define the acceptable patterns of the standard or extended identifier (ID10–ID0 or ID28–ID0). Any of these bits can be marked “don’t care” in the MSCAN identifier mask registers. See Section 22.3.2.17, “MSCAN Identifier Mask Register (CANIDMR0–CANIDMR7).”

A filter hit is indicated to the application software by a set receive buffer full flag (RXF = 1) and 3 bits in the CANIDAC register. See Section 22.3.2.16, “MSCAN Identifier Acceptance Register (CANIDAR0–CANIDAR7).” These identifier hit flags (IDHIT2–0) clearly identify the filter section that caused the acceptance. They simplify the application software’s task to identify the cause of the receiver interrupt. In case more than one hit occurs (two or more filters match), the lower hit has priority.

A flexible programmable generic identifier acceptance filter has been introduced to reduce the Power Architecture interrupt loading. The filter is programmable to operate in four different modes[4]:

- Two identifier acceptance filters, each to be applied to:
  - The full 29 bits of the extended identifier and to the following bits of the CAN 2.0B frame:
    - Remote transmission request (RTR)
    - Identifier extension (IDE)

1. Only if the RXF flag is not set.
2. The receive interrupt occurs only if not masked. A polling scheme can be applied on RXF also.
3. Reference the Bosch CAN 2.0A/B protocol specification dated September 1991 for details.
4. For a better understanding of references made within the filter mode description, reference the Bosch specification dated September 1991 which details the CAN 2.0A/B protocol.

– Substitute remote request (SRR)

— The 11 bits of the standard identifier plus the RTR and IDE bits of the CAN 2.0A/B messages[1]. This mode implements two filters for a full length CAN 2.0B compliant extended identifier. Figure 22-37 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces a filter 0 hit. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces a filter 1 hit.

- Four identifier acceptance filters, each to be applied to
  — a) the 14 most significant bits of the extended identifier plus the SRR and IDE bits of CAN 2.0B messages or
  — b) the 11 bits of the standard identifier, the RTR and IDE bits of CAN 2.0A/B messages. Figure 22-38 shows how the first 32-bit filter bank (CANIDAR0–CANIDA3, CANIDMR0–3CANIDMR) produces filter 0 and 1 hits. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces filter 2 and 3 hits.
- Eight identifier acceptance filters, each to be applied to the first 8 bits of the identifier. This mode implements eight independent filters for the first 8 bits of a CAN 2.0A/B compliant standard identifier or a CAN 2.0B compliant extended identifier. Figure 22-36 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces filter 0 to 3 hits. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces filter 4 to 7 hits.
- Closed filter. No CAN message is copied into the foreground buffer RxFG, and the RXF flag is never set.

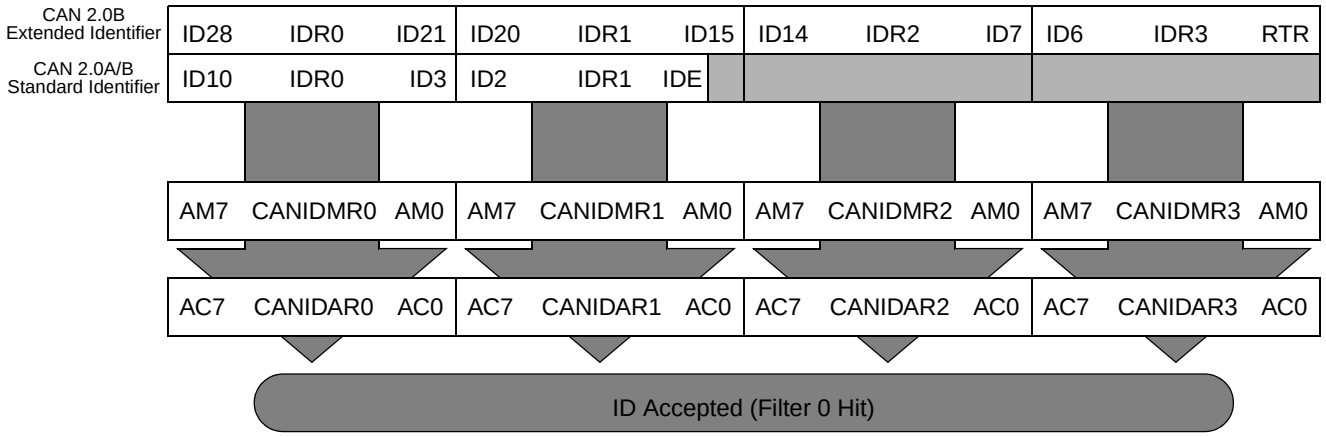


Figure 22-34. 32-Bit Maskable Identifier Acceptance Filter

1.Although this mode can be used for standard identifiers, it is recommended to use the four or eight identifier acceptance filters for standard identifiers

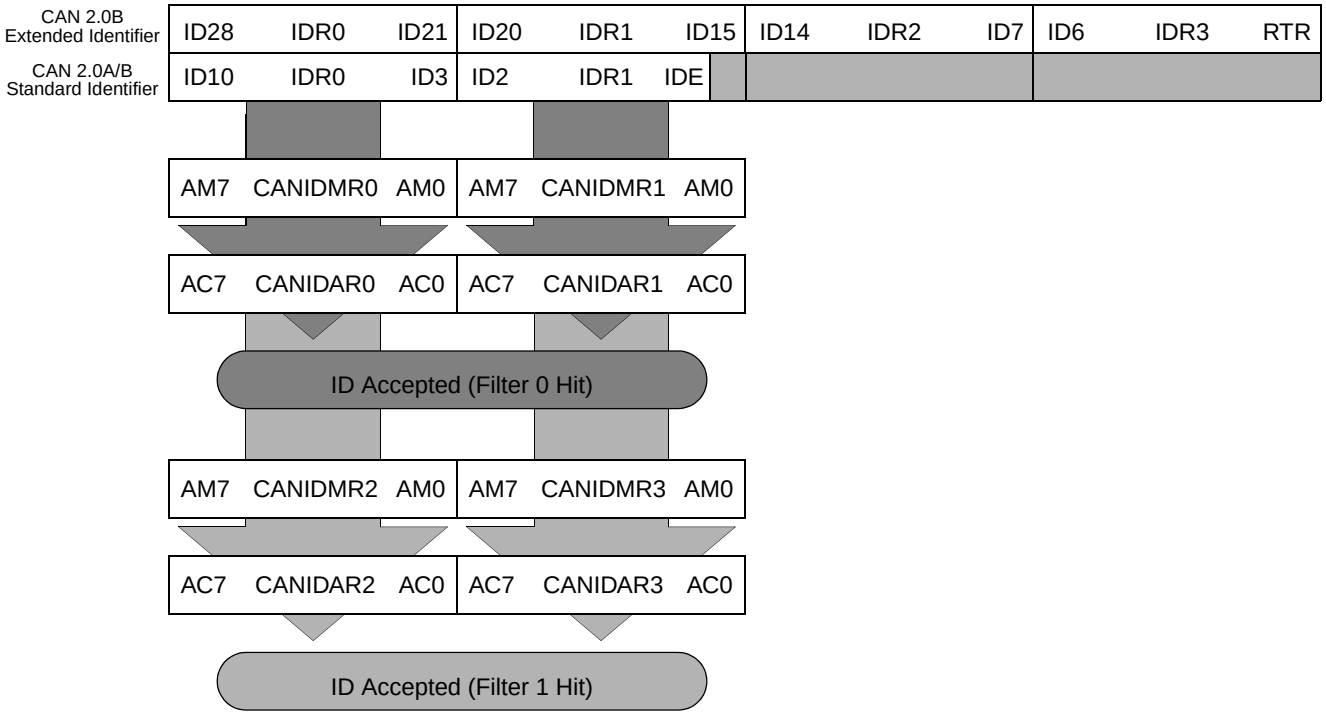


Figure 22-35. 16-Bit Maskable Identifier Acceptance Filters

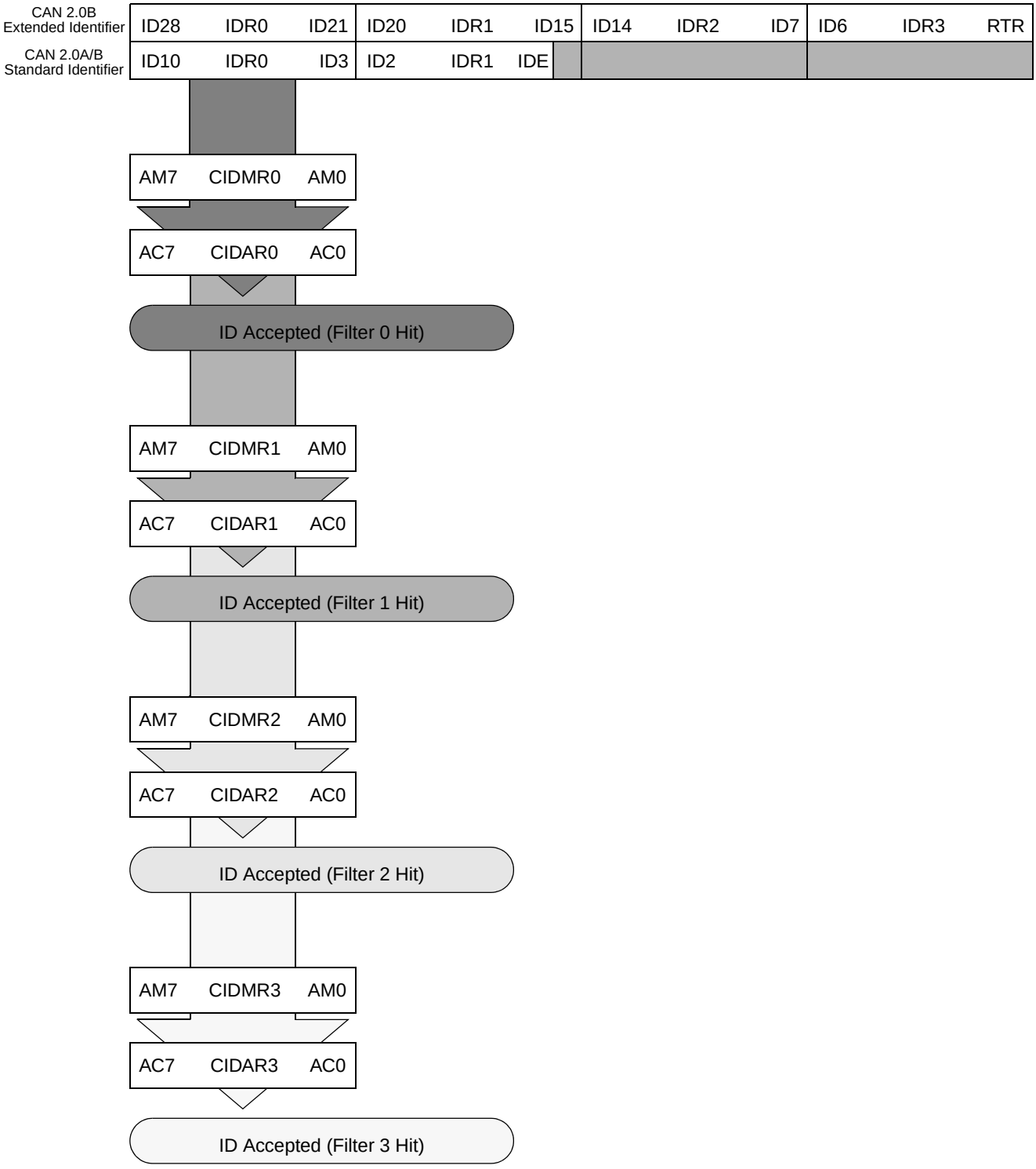


Figure 22-36. 8-Bit Maskable Identifier Acceptance Filters

22.4.4 Protocol Violation Protection

The MSCAN protects you from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters cannot be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN cannot be modified while the MSCAN is on-line. The MSCAN has to be in Initialization Mode. The corresponding INITRQ/INITAK handshake bits in the CANCTL0/CANCTL1 registers (see Section 22.3.2.1, “MSCAN Control 0 Register (CANCTL0)”) serve as a lock to protect the following registers:
  - MSCAN Control 1 Register (CANCTL1)
  - MSCAN Bus Timing Registers 0 and 1 (CANBTR0, CANBTR1)
  - MSCAN Identifier Acceptance Control Register (CANIDAC)
  - MSCAN Identifier Acceptance Registers (CANIDAR0-7)
  - MSCAN Identifier Mask Registers (CANIDMR0-7)
- The TXCAN pin is immediately forced to a recessive state when the MSCAN goes into the power down mode or initialization mode (see Section 22.4.8.3, “MSCAN Power Down Mode,” and Section 22.4.8.2, “MSCAN Initialization Mode”).

22.4.5 Clock System

Figure 22-37 shows the structure of the MSCAN clock generation circuitry. With this flexible clocking scheme, the MSCAN can manage CAN bus rates ranging from 10 kbit/s to 1 Mbit/s.

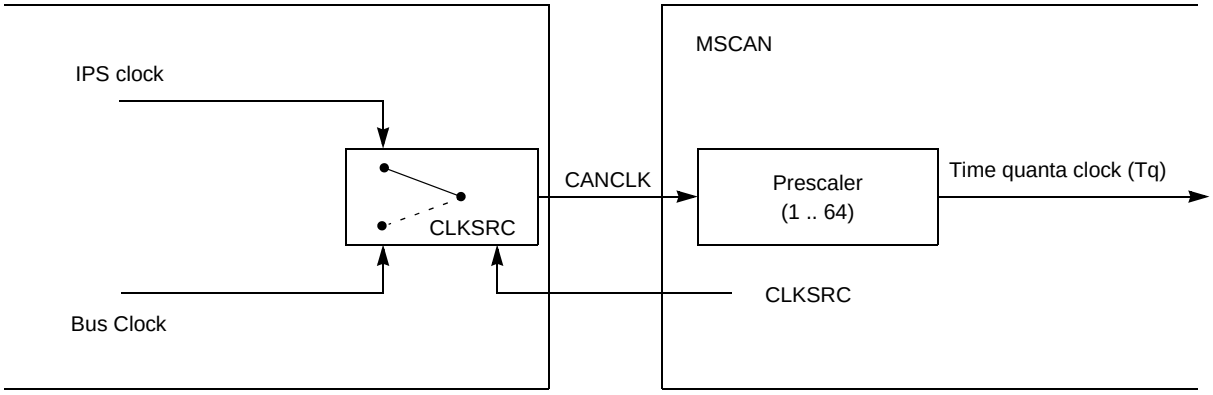


Figure 22-37. MSCAN Clocking Scheme

The clock source bit (CLKSRC) in the CANCTL1 register (see Section 22.3.2.2, “MSCAN Control 1 Register (CANCTL1)”) defines whether the internal CANCLK is connected to the output of the system bus clock or to the IPS clock.

NOTE

Both MSCAN modules can have different selected clock sources. To select the bus clock, the CLKSRC bit in the CANCTL1 register must be set.

The clock source has to be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met. Additionally, for high CAN bus rates (1 Mbit/s), a 45% – 55% duty cycle of the clock is required.

A programmable prescaler generates the time quanta (Tq) clock from CANCLK. A time quantum is the atomic unit of time handled by the MSCAN.

Eqn. 22-2

$$f_{Tq} = \frac{f_{CANCLK}}{(Prescaler \cdot value)}$$

A bit time is subdivided into three segments[1] [2] (see Figure 22-38):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time Segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard.
- Time Segment 2: This segment represents the PHASE_SEG2 of the CAN standard. Setting the parameter TSEG2 to consist of 2 to 8 time quanta long can programmed it.

Eqn. 22-3

$$Bit\ Rate = \frac{f_{Tq}}{(number\ of\ Time\ Quanta)}$$

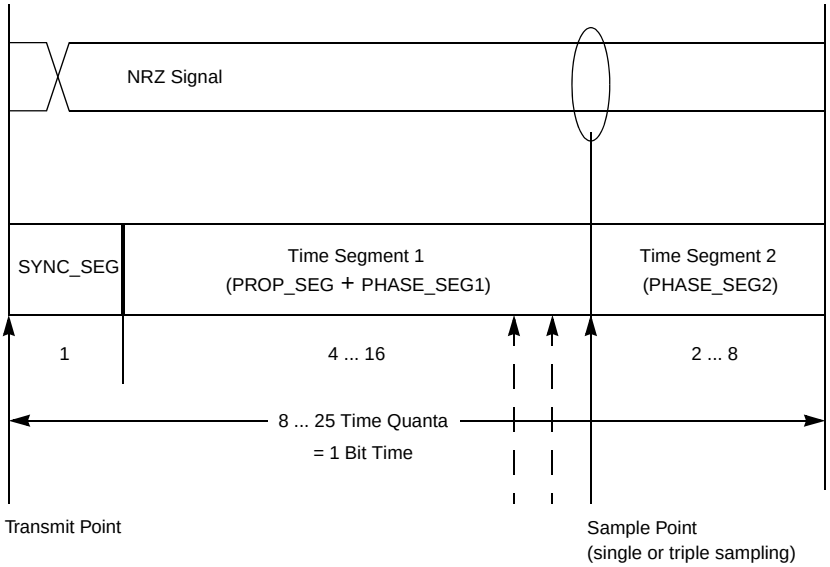


Figure 22-38. Segments within the Bit Time

1. For further explanation of the under-lying concepts, refer to ISO/DIS 11519-1, Section 10.3.
2. Reference the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

Table 22-31. Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node in receive mode samples the bus at this point. If the three samples per bit option is selected, this point marks the position of the third sample.

The synchronization jump width[1] can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

The above parameters are set by programming the MSCAN Bus Timing Registers (CANBTR0, CANBTR1) (see Section 22.3.2.1, “MSCAN Control 0 Register (CANCTL0),” and Section 22.3.2.4, “MSCAN Bus Timing Register 1 (CANBTR1)”).

Table 22-32 gives an overview of the CAN compliant segment settings and the related parameter values.

NOTE

Ensure the bit time settings are in compliance with the CAN standard.

Table 22-32. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	TSEG1	Time Segment 2	TSEG2	Synchronization Jump Width	SJW
5 .. 10	4 .. 9	2	1	1 .. 2	0 .. 1
4 .. 11	3 .. 10	3	2	1 .. 3	0 .. 2
5 .. 12	4 .. 11	4	3	1 .. 4	0 .. 3
6 .. 13	5 .. 12	5	4	1 .. 4	0 .. 3
7 .. 14	6 .. 13	6	5	1 .. 4	0 .. 3
8 .. 15	7 .. 14	7	6	1 .. 4	0 .. 3
9 .. 16	8 .. 15	8	7	1 .. 4	0 .. 3

22.4.6 Timer Link

The MSCAN generates an internal time stamp when a valid frame is received or transmitted and the TIME bit is enabled. Because the CAN specification defines a frame to be valid if no errors occur before the End of Frame (EOF) field is transmitted successfully, the actual value of an internal timer is written at EOF to the appropriate time stamp position within the transmit buffer. For receive frames, the time stamp is written to the receive buffer.

1. Reference the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

22.4.7 Modes of Operation

22.4.7.1 Normal Mode

The MSCAN module behaves as described within this specification in all normal modes.

22.4.7.2 Initialization Mode

The MSCAN is put into initialization mode when INITRQ = 1 and INITAK = 0.

22.4.7.3 Sleep mode

The MSCAN is put into sleep mode When SLPRQ = 1 and SLPAK = 1.

22.4.7.4 Power down mode

The MSCAN is put into power down mode when Power Architecture goes into deep sleep mode.

22.4.7.5 Listen-Only Mode

In an optional bus monitoring mode (listen-only), the CAN node can receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus. In addition, it cannot start a transmission. If the MAC sub-layer is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so the MAC sub-layer monitors this dominant bit; although, the CAN bus may remain in recessive state externally.

22.4.8 Low Power Options

If the MSCAN is disabled (CANE = 0), the MSCAN clocks are stopped for power savings.

If the MSCAN is enabled (CANE = 1), the MSCAN has two additional modes with reduced power consumption, compared to normal mode: sleep and power down mode. In sleep mode, power consumption is reduced by stopping all clocks except those to access the registers from the Power Architecture side. In power down mode, all clocks are stopped and no power is consumed.

Table 22-33 summarizes the MSCAN modes. A particular combination of modes is entered by the given settings on the SLPRQ/SLPAK bits.

For all modes, an MSCAN wake-up interrupt can only occur if the MSCAN is in sleep mode (SLPRQ = 1 and SLPAK = 1), wake-up functionality is enabled (WUPE = 1), and the wake-up interrupt is enabled (WUPIE = 1).

Table 22-33. MSCAN Operating Modes

MSCAN Mode			
Normal	**Power Down (Power Architecture enters deep sleep)**	**Sleep**	**(CANE = 0)**
SLPRQ = 0 SLPAK = 0	SLPRQ = X SLPAK = X	SLPRQ = 1 SLPAK = 1	SLPRQ = X[1] SLPAK = X

1 X means “don’t care”

22.4.8.1 MSCAN Sleep Mode

The Power Architecture can request the MSCAN to enter this low power mode by asserting the SLPRQ bit in the CANCTL0 register. The time when the MSCAN enters sleep mode depends on a fixed-synchronization delay and its current activity:

- If there are one or more message buffers scheduled for transmission (TXEx = 0), the MSCAN continues to transmit until all transmit message buffers are empty (TXEx = 1, transmitted successfully or aborted) and then goes into sleep mode.
- If the MSCAN is receiving, it continues to receive and goes into sleep mode as soon as the CAN bus next becomes idle.
- If the MSCAN is neither transmitting nor receiving, it immediately goes into sleep mode.

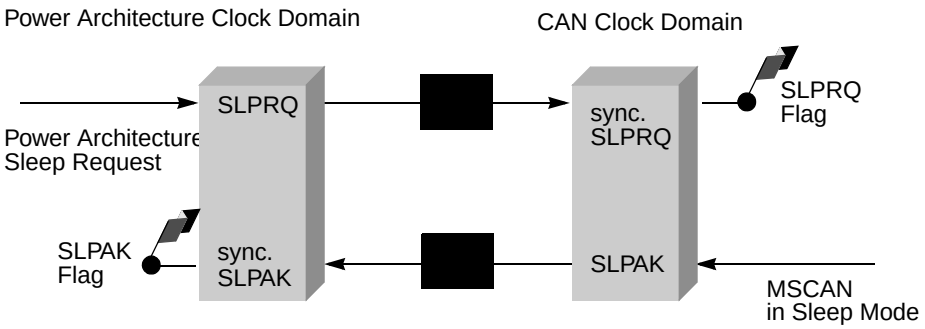


Figure 22-39. Sleep Request/Acknowledge Cycle

NOTE

The application software must avoid setting up a transmission (by clearing one or more TXEx flag[s]) and immediately request sleep mode (by setting SLPRQ). It depends on the exact sequence of operations whether the MSCAN starts transmitting or goes into sleep mode directly.

If sleep mode is active, the SLPRQ and SLPAK bits are set (Figure 22-39). The application software must use SLPAK as a handshake indication for the request (SLPRQ) to go into sleep mode.

When in sleep mode (SLPRQ = 1 and SLPAK = 1), the MSCAN stops its internal clocks. However, clocks that allow register accesses from the Power Architecture side continue to run.

If the MSCAN is in bus-off state, it stops counting the 128 occurrences of 11 consecutive recessive bits due to the stopped clocks. The TXCAN pin remains in a recessive state. If RXF = 1, the message can be read and RXF can be cleared. Shifting a new message into the foreground buffer of the receiver FIFO (RxFG) does not take place while in sleep mode.

It is possible to access the transmit buffers and to clear the associated TXE flags. No message abort takes place while in sleep mode.

If the WUPE bit in CANCLT0 is not asserted, the MSCAN masks any activity it detects on CAN. The RXCAN pin is therefore held internally in a recessive state. This locks the MSCAN in sleep mode.The MSCAN is able to leave sleep mode (wake up) only when:

- CAN bus activity occurs and WUPE = 1

  or
- the Power Architecture clears the SLPRQ bit

NOTE

The Power Architecture cannot clear the SLPRQ bit before sleep mode (SLPRQ = 1 and SLPAK = 1) is active.After wake-up, the MSCAN waits for 11 consecutive recessive bits to synchronize to the bus. As a consequence, if the MSCAN is woken-up by a CAN frame, this frame is not received. The receive message buffers (RxFG and RxBG) contain messages if they were received before sleep mode was entered. All pending actions are executed upon wake-up; copying of RxBG into RxFG, message aborts and message transmissions. If the MSCAN remains in bus-off state after sleep mode was left, it continues counting the 128 × 11 consecutive recessive bits.

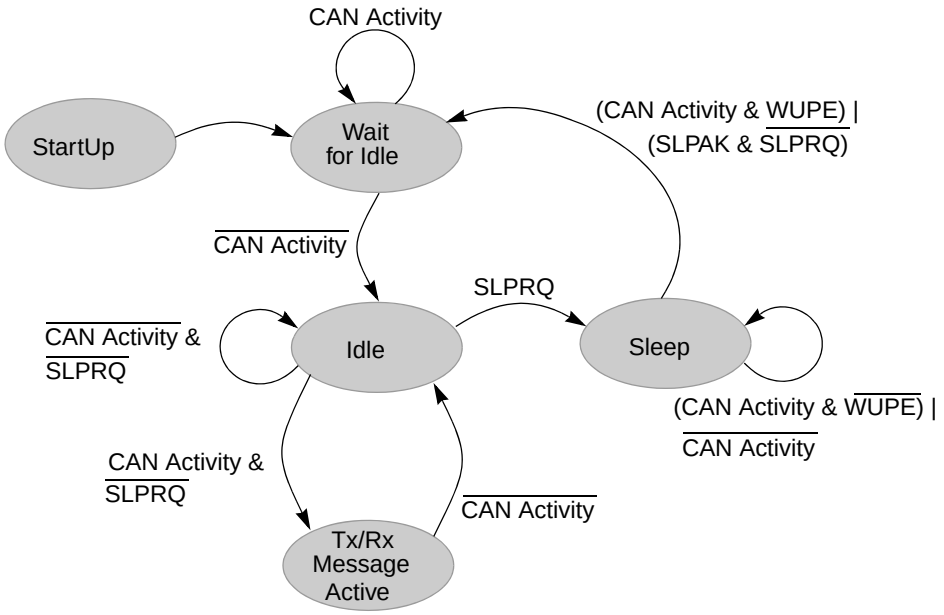


Figure 22-40. Simplified State Transitions for Entering/Leaving Sleep Mode

22.4.8.2 MSCAN Initialization Mode

In initialization mode, any ongoing transmission or reception is immediately aborted and synchronization to the bus is lost potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations, the MSCAN immediately drives the TXCAN pin into a recessive state.

NOTE

Ensure the MSCAN is not active when initialization mode is entered. The recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPAK = 1) before setting the INITRQ bit in the CANCTL0 register. Otherwise, the abort of an ongoing message can cause an error condition and can have an impact on the other bus devices.

In initialization mode, the MSCAN is stopped. However, interface registers can continue to be accessed. This mode resets the CANTCTL0, CANRFLG, CANRIER, CANTFLG, CANTIER, CANTARQ, CANTAAK, and CANTBSEL registers to their default values. In addition, it enables the configuration of the CANBTR0 and CANBTR1 bit timing registers, CANIDAC, and the CANIDAR and CANIDMR message filters. See Section 22.3.2.1, “MSCAN Control 0 Register (CANCTL0),” for a detailed description of the initialization mode.

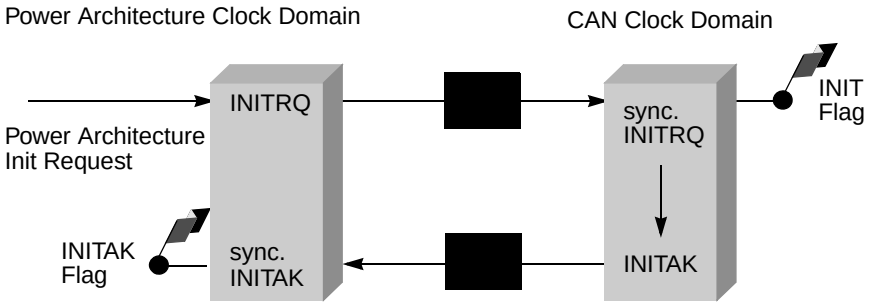


Figure 22-41. Initialization Request/Acknowledge Cycle

Due to independent clock domains within the MSCAN, INITRQ must be synchronized to all domains by using a special handshake mechanism. This handshake causes additional synchronization delay (see Section Figure 22-41., “Initialization Request/Acknowledge Cycle”).

If there is no message transfer ongoing on the CAN bus, the minimum delay is two additional bus clocks and three additional CAN clocks. When all parts of the MSCAN are in initialization mode, the INITAK flag is set. The application software must use INITAK as a handshake indication for the request (INITRQ) to go into initialization mode.

NOTE

The Power Architecture cannot clear the INITRQ bit before initialization mode (INITRQ = 1 and INITAK = 1) is active.

22.4.8.3 MSCAN Power Down Mode

The MSCAN is in power down mode (Table 22-33) when the Power Architecture is in deep sleep mode

When entering the power down mode, the MSCAN immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations to the above rule, the MSCAN immediately drives the TXCAN pin into a recessive state.

NOTE

You are responsible for ensuring that the MSCAN is not active when Power Architecture deep sleep mode is entered. The recommended procedure is to bring the MSCAN into Sleep mode before the Power Architecture enters deep sleep mode. Otherwise, the abort of an ongoing message can cause an error condition and impact other CAN bus devices.

In power down mode, all clocks are stopped and no registers can be accessed. If the MSCAN was not in sleep mode before power down mode became active, the module would perform an internal recovery cycle after powering up. This causes some fixed delay before the module enters run mode again.

22.4.8.4 Programmable Wake-Up Function

The MSCAN can be programmed to wake-up the MSCAN as soon as bus activity is detected (see control bit WUPE in Section 22.3.2.1, “MSCAN Control 0 Register (CANCTL0)”).

22.4.9 Reset Initialization

The reset state of each individual bit is listed in Section 22.3.2, “Register Descriptions,” which details all the registers and their bit-fields.

22.4.10 Interrupts

This section describes all interrupts originated by the MSCAN. It documents the enable bits and generated flags. Each interrupt is listed and described separately.

22.4.11 Description of Interrupt Operation

The MSCAN supports one interrupt vector mapped onto eight different interrupt sources, any of which can be individually masked (for details, see sections Section 22.3.2.6, “MSCAN Receiver Interrupt Enable Register (CANRIER),” to Section 22.3.2.8, “MSCAN Transmitter Interrupt Enable Register (CANTIER)”).

NOTE

The dedicated interrupt vector addresses are defined in Chapter 17, “Integrated Programmable Interrupt Controller (IPIC).”

Table 22-34. Interrupt Vectors

Interrupt Source	CCR Mask	Local Enable
Wake-Up Interrupt (WUPIF)	I bit	CANRIER (WUPIE)
Error Interrupts Interrupt (CSCIF, OVRIF)	I bit	CANRIER (CSCIE, OVRIE)

Table 22-34. Interrupt Vectors (Continued)

Interrupt Source	CCR Mask	Local Enable
Receive Interrupt (RXF)	I bit	CANRIER (RXFIE)
Transmit Interrupts (TXE[2:0])	I bit	CANTIER (TXEIE[2:0])

22.4.11.1 Transmit Interrupt

At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXEx flag of the empty message buffer is set.

22.4.11.2 Receive Interrupt

A message is successfully received and shifted into the foreground buffer (RxFG) of the receiver FIFO. This interrupt is generated immediately after receiving the EOF symbol. The RXF flag is set. If there are multiple messages in the receiver FIFO, the RXF flag is set as soon as the next message is shifted to the foreground buffer.

22.4.11.3 Wake-Up Interrupt

Activity on the CAN bus occurred during MSCAN internal sleep mode and WUPE (see Section 22.3.2.1, “MSCAN Control 0 Register (CANCTL0)”) enabled.

22.4.11.4 Error Interrupt

An error interrupt is generated if an overrun of the receiver FIFO, error, warning, or bus-off condition occurs. See Section 22.3.2.5, “MSCAN Receiver Flag Register (CANRFLG).” An interrupt indicates one of the following conditions:

- Overrun – An overrun condition of the receiver FIFO as described in Section 22.4.2.3, “Receive Structures,” occurred.
- CAN Status Change – The actual value of the transmit and receive error counters control the CAN bus state of the MSCAN. As soon as the error counters skip into a critical range (Tx/Rx-warning, Tx/Rx-error, bus-off) the MSCAN flags an error condition. The status change that caused the error condition is indicated by the TSTAT and RSTAT flags (see Section 22.3.2.5, “MSCAN Receiver Flag Register (CANRFLG),” and Section 22.3.2.6, “MSCAN Receiver Interrupt Enable Register (CANRIER)”).

22.4.12 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the CANRFLG register (see Section 22.3.2.5, “MSCAN Receiver Flag Register (CANRFLG)”) or the CANTFLG register (see Section 22.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”). Interrupts are pending as long as one of the corresponding flags is set. The flags in these registers must be reset within the interrupt manager to handshake the interrupt. The flags are reset by writing a 1 to the corresponding bit position.

NOTE

It must be guaranteed that the Power Architecture only clears the bit causing the current interrupt. RXFIF interrupt clear has the side effect of moving the receive buffer from the current to the next. Clearing a TXE*x*IF flag also clears the corresponding ABTAK*x*. When a TXE*x*IF flag is set, the corresponding ABTRQ*x* bit is cleared

22.4.13 Recovery from Deep Sleep Mode

The MSCAN can wake up the Power Architecture from deep sleep mode via the wake-up interrupt. This interrupt can only occur if the MSCAN is in Sleep Mode (SLPRQ = 1 and SLPAK = 1), the wake-up option is enabled (WUPE = 1), and the wake-up interrupt is enabled (WUPIE = 1).

When there is recessive to dominant change in CAN_RX pin, there is a wake-up interrupt generated asynchronously. Power Architecture can then be waken up from deep sleep mode by that interrupt.

After Power Architecture be waken up from deep sleep mode, it can enable the clocks of MSCAN and put MSCAN into normal operation mode again.

NOTE

Only MSCAN modules 1 and 2 can wake up the MPC5125 from deep sleep. MSCAN modules 3 and 4 cannot wake up the MPC5125 from deep sleep mode.

22.4.14 MSCAN Initialization

The procedure to initially start up the MSCAN module out of reset is as follows:

1. Assert CANE.
2. Write to the configuration registers in initialization mode.
3. Clear INITRQ to leave initialization mode and enter normal mode.

If the configuration of the registers that are writable in initialization mode only needs to be changed when the MSCAN module is in normal mode, do the following:

1. Make sure that the MSCAN transmission queue becomes empty and bring the module into sleep mode by asserting SLPRQ and awaiting SLPAK.
2. Enter initialization mode: Assert INITRQ and await INITAK.
3. Write to the configuration registers in initialization mode.
4. Clear INITRQ to leave initialization mode and continue in normal mode.

22.4.15 Bus-Off Recovery

The bus-off recovery is configurable: the bus-off state can be left automatically or on request.

For reasons of backwards compatibility the MSCAN defaults to automatic recovery after reset. In this case, the MSCAN becomes error active again after counting 128 occurrences of 11 consecutive recessive bits on the CAN bus. These two events may occur in any order

This page is intentionally left blank.

Chapter 23
NAND Flash Controller (NFC)

23.1 Introduction

Composed of various control logic units and a 9 KB SRAM buffer, the NAND Flash Controller (NFC) implements the interface to standard NAND flash memory devices. See Figure 23-1 for a block diagram of the NFC.

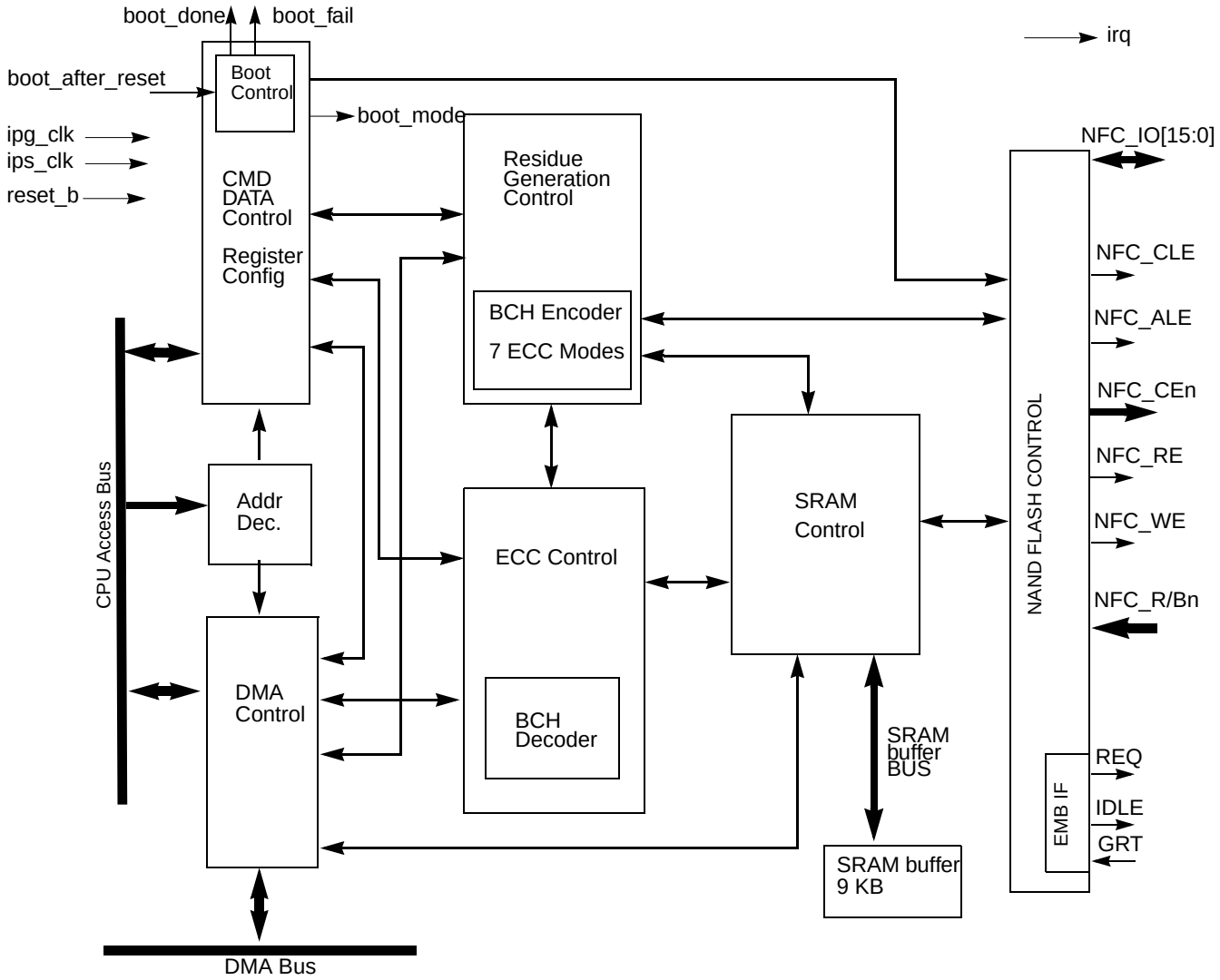


Figure 23-1. NFC Block Diagram

23.2 Overview

The NAND Flash Controller (NFC) interfaces standard NAND flash devices to the IC and hides the complexities of accessing the NAND flash. It provides a glueless interface to both 8-bit and 16-bit NAND flash parts with page sizes of 512 bytes, 2 KB, 4 KB, and 8 KB.

23.3 Features

The NAND flash controller includes the following features:

- Organization
  - NAND flash interface: 8- or 16-bit
- 9 KB RAM buffer
  - Memory mapped registers and SRAM buffer
- Interface with NAND flash through NFC
  - Supports all NAND flash products regardless of density/organization
  - Supports flash device commands like page read, page program, reset, block erase, read status, read id, copy-back, multiplane read/program, interleaved read/program, random input/output, read in EDO mode, but not limited.
- DMA
  - Integrated DMA engine
  - Two configurable DMA channels
    - Use DMA channel 1 only to read/write a page for both main and spare area of a page
    - Use DMA channel 1 to read/write the main area of a page, and DMA channel 2 for the spare area
- ECC mode / Bypass ECC
  - In ECC mode, NFC supports 4-, 6-, 8-, 12-, 16-, 24-, or 32-bit error correction.
  - ECC mode can be bypassed.
- Boot
  - Boot from page size ≥ 2 KB flash (×8/×16) without extra control
- I/O pins sharing support
  - Allow sharing of the I/O pins with other memory controllers through special arbitration logic.

23.4 External Signal Description

This section describes the NFC external signals.

23.4.1 Overview

The following signals shown in Table 23-1 are used to control the NAND flash device.

Table 23-1. NFC Signal Properties

Name	Function	I/O	Reset
NFC_ALE	Flash Address Latch Enable	O	0
NFC_CE0	Flash #0 Chip Enable	O	1
NFC_CE1	Flash #1 Chip Enable	O	1
NFC_CE2	Flash #2 Chip Enable	O	1
NFC_CE3	Flash #3 Chip Enable	O	1
NFC_CLE	Flash Command Latch Enable	O	0
NFC_R/$\overline{B}$0	Flash #0 Ready/Busy	I	—
NFC_R/$\overline{B}$1	Flash #1 Ready/Busy	I	—
NFC_R/$\overline{B}$2	Flash #2 Ready/Busy	I	—
NFC_R/$\overline{B}$3	Flash #3 Ready/Busy	I	—
NFC_RE	Flash Read Enable	O	1
NFC_WE	Flash Write Enable	O	1
NFC_IO[15:0]	Flash data bus	I/O	—

23.5 NFC Buffer Memory Space

Figure 23-2 shows the organization of the buffer memory space in the NFC. The memory size is 1152 × 64 bits, and is separated into 4 buffers. Each buffer has a non-contiguous physical address. For example, buffer0's physical address is (0x0000 + 0x0020 × i) ~ (0x0007 + 0x0020 × i).

When the CPU is writing or reading a buffer in no- boot mode, the CPU address is continuous, because an address transition is provided inside the NFC:
sram_physical_addr[13:3] = {cpu_addr[11:3],cpu_addr[13:12]}.

So in non-boot mode, buffer0's address range is 0x0000–0x08FF, buffer1 is 0x1000–0x18FF, buffer2 is 0x2000–0x28FF, buffer3 is 0x3000–0x38FF.

See Figure 23-25 for the different operations between boot mode and non-boot mode.

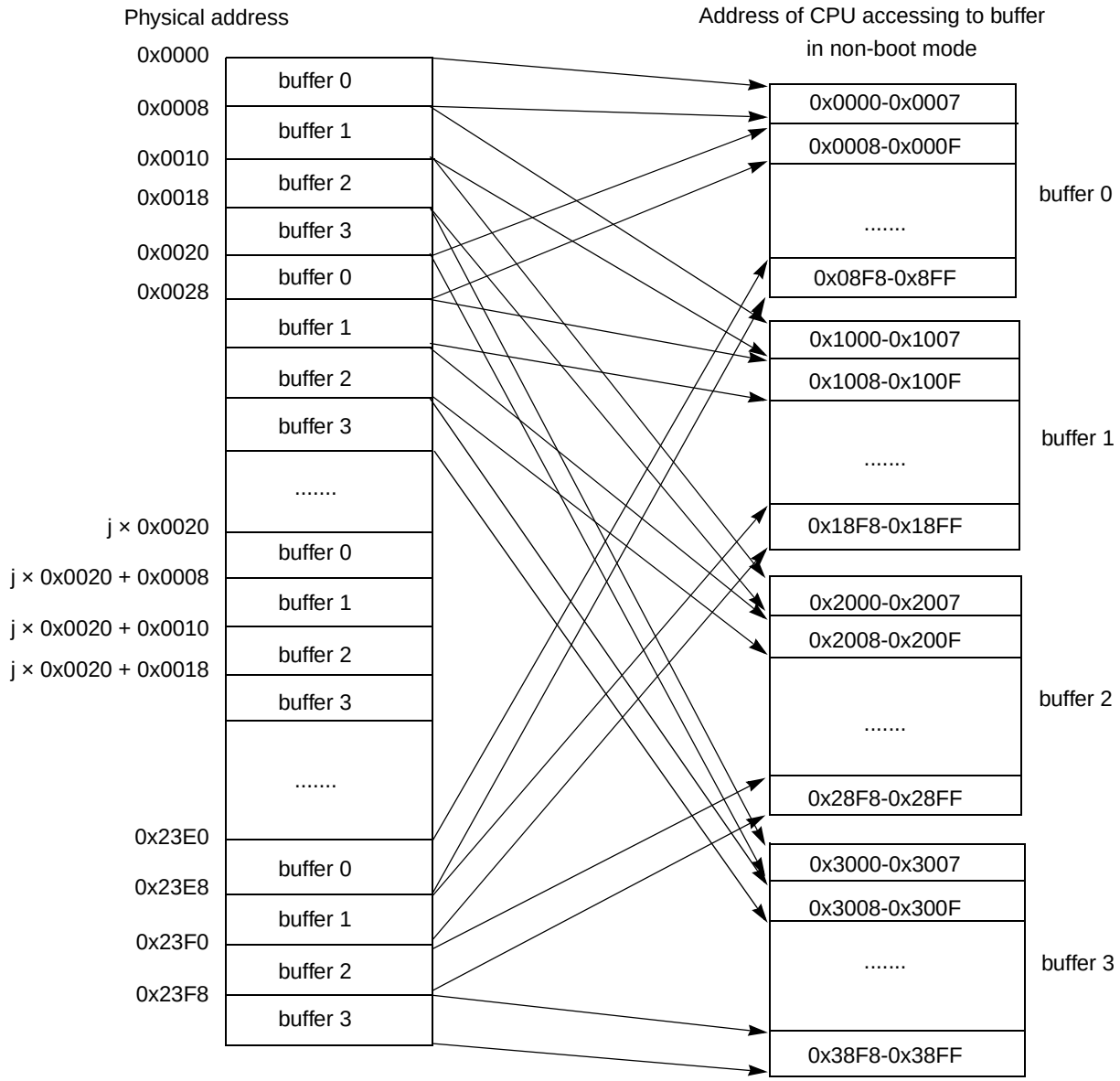


Figure 23-2. Data (Buffer) Organization in Memory

23.6 Memory Map and Register Definition

Section 23.7, "Register Descriptions," provides detailed descriptions for all of the NFC registers.

23.6.1 Memory Map

Table 23-2 shows the NFC memory map.

Table 23-2. NFC memory map

Offset from NFC_BASE (0xFF40_C000)[1]	Register	Access	Reset Value	Section/Page
0x0000–0x38FF	SRAM buffer	R/W	—[2]	23.5/23-613
0x3900–0x3EFF	Reserved			
0x3F00	FLASH_CMD1—Flash Command 1 register	R/W	0x30FE_0000	23.7.1/23-618
0x3F04	FLASH_CMD2—Flash Command 2 register	R/W	0x007E_E001	23.7.2/23-618
0x3F08	COL_ADDR—Column Address register	R/W	0x0000_0000	23.7.3/23-619
0x3F0C	ROW_ADDR—Row Address register	R/W	0x1100_0000	23.7.4/23-620
0x3F10	FLASH_COMMAND_REPEAT—Flash Command Repeat register	R/W	0x0000_0000	23.7.5/23-620
0x3F14	ROW_ADDR_INC—Row Address Increment register	R/W	0x0000_0001	23.7.6/23-621
0x3F18	FLASH_STATUS1—Flash Status 1 register	R	0x0000_0000	23.7.7/23-622
0x3F1C	FLASH_STATUS2—Flash Status 2 register	R	0x0000_0000	23.7.8/23-622
0x3F20	DMA1_ADDR—DMA1 Address register	R/W	0x0000_0000	23.7.9/23-623
0x3F24	DMA_CONFIG—DMA Configuration register	R/W	0x0000_0000	23.7.10/23-623
0x3F28	CACHE_SWAP—Cache Swap register	R/W	0x0FFE_0FFE	23.7.11/23-624
0x3F2C	SECTOR_SIZE—Sector Size register	R/W	0x0000_0420	23.7.12/23-625
0x3F30	FLASH_CONFIG—Flash Configuration register	R/W	0x000E_A631	23.7.13/23-625
0x3F34	DMA2_ADDR—DMA2 Address register	R/W	0x0000_0000	23.7.14/23-627
0x3F38	IRQ_STATUS—IRQ and Status register	R/W	0x0400_0000	23.7.15/23-628
0x0x3F3C–0x3FFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 Reset value is indeterminate.

23.6.2 Register Summary

Table 23-3. NFC Register Summary

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FLASH_CMD1 (0x3F00)	R	FLASH_CMD_BYTE2								FLASH_CMD_BYTE3							
	W																
	R																
	W																
FLASH_CMD2 (0x3F04)	R	FLASH_CMD_BYTE1								FLASH_CMD_CODE							
	W																
	R	FLASH_CMD_CODE												BUFNO			BUSY
	W																START
COL_ADDR (0x3F08)	R																
	W																
	R	COL_ADDR2								COL_ADDR1							
	W																
ROW_ADDR (0x3F0C)	R	CHIP_SEL_RB				CHIP_SEL				ROW_ADDR3							
	W																
	R	ROW_ADDR2								ROW_ADDR1							
	W																
FLASH_COMMAND_REPEAT (0x3F10)	R																
	W																
	R	REPEAT_COUNT															
	W																
ROW_ADDR_INC (0x3F14)	R									ROW_ADDR3_INC							
	W																
	R	ROW_ADDR2_INC								ROW_ADDR1_INC							
	W																
FLASH_STATUS1 (0x3F18)	R	ID_BYTE1								ID_BYTE2							
	W																
	R	ID_BYTE3								ID_BYTE4							
	W																
FLASH_STATUS2 (0x3F1C)	R	ID_BYTE5															
	W																
	R									STATUS_BYTE1							
	W																

Table 23-3. NFC Register Summary (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DMA1_ADDR (0x3F20)	R	DMA1_ADDR															
	W																
	R																
	W																
DMA2_ADDR (0x3F34)	R	DMA2_ADDR															
	W																
	R																
	W																
DMA_CONFIG (0x3F24)	R	DMA1_CNT												DMA2_CNT			
	W																
	R	DMA2_CNT				DMA2_OFFSET										DMA1_ACT	DMA2_CNT
	W																
CACHE_SWAP (0x3F28)	R					CACHE_SWAP_ADDR2											
	W																
	R					CACHE_SWAP_ADDR1											
	W																
SECTOR_SIZE (0x3F30)	R																
	W																
	R					SECTOR_SIZE											
	W																
FLASH_CONFIG (0x3F30)	R	STOP_ON_WERR	ECC_SRAM_ADDR									ECC_SRAM_REQ	DMA_REQ	ECC_MODE			FAST_FLASH
	W																
	R	ID_COUNT			CMD_TIMEOUT					16 BIT	BOOT_MODE	ADDR_AUTO_INCR	BUFNO_AUTO_INCR	PAGE_CNT			
	W																
IRQ_STATUS (0x3F38)	R	WERR_IRQ	CMD_DONE_IRQ	IDLE_IRQ		WERR_STATUS	FLASH_CMD_BUSY	RESIDUE_BUSY	ECC_BUSY	DMA_BUSY	WERR_EN	CMD_DONE_EN	IDLE_EN				
	W													WERR_CLEAR	CMD_DONE_CLEAR	IDLE_CLEAR	
	R											RESIDUE_BUFF_NO		ECC_BUFF_NO		DMA_BUFF_NO	
	W																

23.7 Register Descriptions

23.7.1 Flash Command 1 register (FLASH_CMD1)

Address: Base +0x3F00 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FLASH_CMD_BYTE2[7:0]								FLASH_CMD_BYTE3[7:0]							
Reset	0	0	1	1	0	0	0	0	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-3. Flash Command 1 register (FLASH_CMD1)

Table 23-4. FLASH_CMD1 field descriptions

Field	Description
FLASH_CMD_BYTE2[7:0]	Second command byte that may be sent to flash device.
FLASH_CMD_BYTE3[7:0]	Third command byte that may be sent to flash device.

23.7.2 Flash Command 2 register (FLASH_CMD2)

Address: Base + 0x3F04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FLASH_CMD_BYTE1[7:0]								FLASH_CMD_CODE[15:8]							
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLASH_CMD_CODE[7:0]								0	0	0	0	0	BUFNO		BUSY
W																START
Reset	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 23-4. Flash Command 2 register (FLASH_CMD2)

Table 23-5. FLASH_CMD2 field descriptions

Field	Description
FLASH_CMD_BYTE1[7:0]	First command byte that may be sent to flash device.
FLASH_CMD_CODE[15:0]	User defined flash operation sequencer. Each bit stands for a certain action.(See Table 23-22) If the bit is set to 1, the corresponding action will be executed, after writing 1 to START field. Following are some configuration examples: (other sequences are possible.) 0111_1110_1110_0000 : read data (FLASH_CMD_BYTE1, 5x Address, FLASH_CMD_BYTE2, R/$\overline{B}$, read data). 1111_1111_1101_1000 : write page (DMA,FLASH_CMD_BYTE1, 5x Address, write data, FLASH_CMD_BYTE2, R/$\overline{B}$, FLASH_CMD_BYTE3, read status). 0100_1110_1101_1000 : block erase (FLASH_CMD_BYTE1, 3x Address, FLASH_CMD_BYTE2, R/$\overline{B}$, FLASH_CMD_BYTE3, read status). 0100_1000_0000_0100 : Read ID (FLASH_CMD_BYTE1, 1x Address, read ID). 0100_0000_0100_0000 : Reset (FLASH_CMD_BYTE1, R/$\overline{B}$). 0111_1110_0000_0000 : CMD+address (FLASH_CMD_BYTE1, 5x Address). 1111_1111_1100_0000 : write page burst (DMA,FLASH_CMD_BYTE1,5xAddress, write data, FLASH_CMD_BYTE2,R/$\overline{B}$).
BUFNO[1:0]	Internal buffer number used for this command.
BUSY	This bit is repeated in the interrupt register. 0 Flash controller is idle. OK to sent next command. 1 Command execution is busy. **Note:** The reset value is maintained a short period of time. The NFC issues the reset command 0xFF to NAND flash after system reset. After the command is done, the value of BUSY changes to 0.
START	0 No action. 1 Command execution starts.

23.7.3 Column Address register (COL_ADDR)

Address: Base + 0x3F08 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COL_ADDR2[7:0]								COL_ADDR1[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-5. Column Address register (COL_ADDR)

Table 23-6. COL_ADDR field descriptions

Field	Description
COL_ADDR1[7:0]	First byte of column address.
COL_ADDR2[7:0]	Second byte of column address.

23.7.4 Row Address register (ROW_ADDR)

Address: Base + 0x3F0C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CHIP_SEL_RB[3:0]				CHIP_SEL[3:0]				ROW_ADDR3[7:0]							
Reset	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ROW_ADDR2[7:0]								ROW_ADDR1[7:0]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-6. Row Address register (ROW_ADDR)

Table 23-7. ROW_ADDR field descriptions

Field	Description
CHIP_SEL[3:0]	Chip select - Bit mask to enable/disable each of the four chip selects. 0 Corresponding $\overline{\text{NFC\_CE}}$ is disabled. 1 Corresponding $\overline{\text{NFC\_CE}}$ is enabled.
CHIP_SEL_RB[3:0]	The field CHIP_SEL determines which $\overline{\text{NFC\_CE}}$ line will go low, the field CHIP_SEL_RB determines to which NFC_R/$\overline{\text{B}}$ lines will be waited on a wait for R/$\overline{\text{B}}$ command. Normally, when 4 $\overline{\text{NFC\_CE}}$ lines and 4 NFC_R/$\overline{\text{B}}$ lines are used, the two fields must contain identical values. When 4 $\overline{\text{NFC\_CE}}$ lines, and 1 NFC_R/$\overline{\text{B}}$ line is used, then CHIP_SEL is the true chip select, and CHIP_SEL_RB is always 1.
ROW_ADDR1[7:0]	First byte of row address.
ROW_ADDR2[7:0]	Second byte of row address.
ROW_ADDR3[7:0]	Third byte of row address.

23.7.5 Flash Command Repeat register (FLASH_COMMAND_REPEAT)

Address: Base + 0x3F10 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	REPEAT_COUNT[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-7. Flash Command Repeat register (FLASH_COMMAND_REPEAT)

Table 23-8. FLASH_COMMAND_REPEAT field descriptions

Field	Description
REPEAT_COUNT[15:0]	16-bit repeat count. It determines how many times FLASH_CMD_CODE in FLASH_CMD2 is executed. See Section 23.7.2, “Flash Command 2 register (FLASH_CMD2).” If 0 or 1, flash command is executed once.

23.7.6 Row Address Increment register (ROW_ADDR_INC)

Address: Base + 0x3F14 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	ROW_ADDR3_INC[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ROW_ADDR2_INC[7:0]								ROW_ADDR1_INC[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-8. Row Address Increment register (ROW_ADDR_INC)

Table 23-9. ROW_ADDR_INC field descriptions

Field	Description
ROW_ADDR1_INC[7:0]	Increment for the first byte of row address.
ROW_ADDR2_INC[7:0]	Increment for the second byte of row address.
ROW_ADDR3_INC[7:0]	Increment for the third byte of row address.

Note: When auto-increment of row address is enabled (ADDR_AUTO_INCR = 1, Section 23.7.13, “Flash Configuration register (FLASH_CONFIG)”), row address is incremented as follows:

new{ROW_ADDR3, ROW_ADDR2, ROW_ADDR1} = {ROW_ADDR3, ROW_ADDR2, ROW_ADDR1} + {ROW_ADDR3_INC, ROW_ADDR2_INC, ROW_ADDR1_INC}

23.7.7 Flash Status 1 register (FLASH_STATUS1)

Address: Base + 0x3F18 Access: User read-only

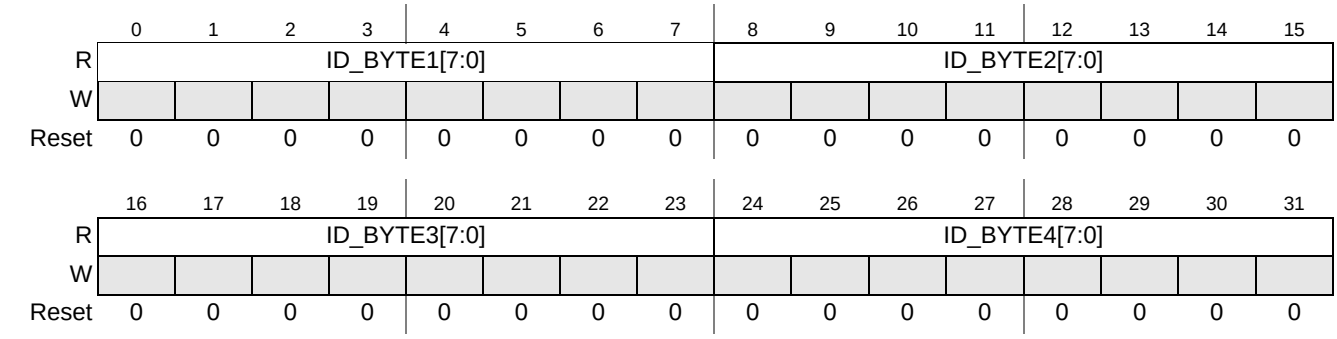


Figure 23-9. Flash Status 1 register (FLASH_STATUS1)

Table 23-10. FLASH_STATUS1 field descriptions

Field	Description
ID_BYTE1[7:0]	First byte returned by *read ID* command
ID_BYTE2[7:0]	Second byte returned by *read ID* command
ID_BYTE3[7:0]	Third byte returned by *read ID* command
ID_BYTE4[7:0]	Fourth byte returned by *read ID* command

23.7.8 Flash Status 2 register (FLASH_STATUS2)

Address: Base + 0x3F1C Access: User read-only

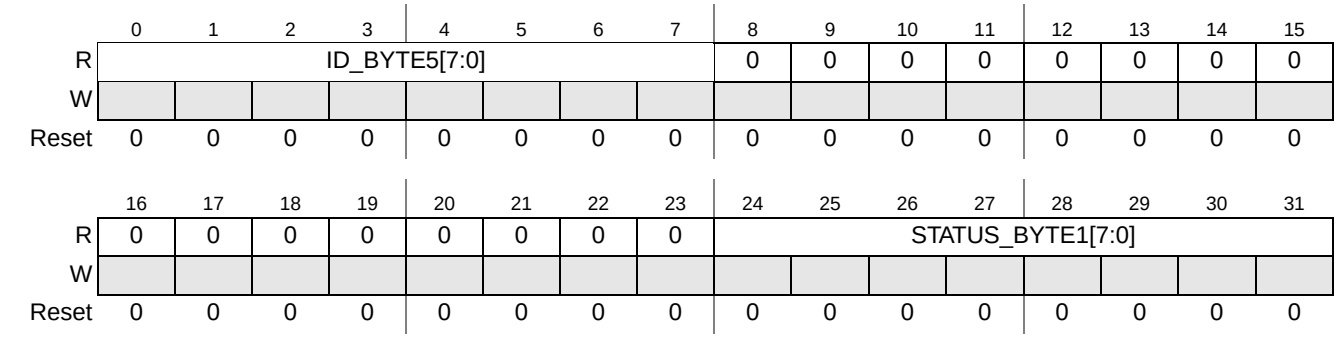


Figure 23-10. Flash Status 2 register (FLASH_STATUS2)

Table 23-11. FLASH_STATUS2 field descriptions

Field	Description
ID_BYTE5[7:0]	Fifth byte returned by *read ID* command
STATUS_BYTE1[7:0]	Byte returned by *read status* command

23.7.9 DMA1 Address register (DMA1_ADDR)

Address: Base + 0x3F20 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DMA1_ADDR[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DMA1_ADDR[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-11. DMA1 Address register (DMA1_ADDR)

Table 23-12. DMA1_ADDR field descriptions

Field	Description
DMA1_ADDR[31:0]	DMA channel 1 address

23.7.10 DMA Configuration register (DMA_CONFIG)

Address: Base + 0x3F24 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DMA1_CNT[11:0]												DMA2_CNT[6:3]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DMA2_CNT[2:0]				DMA2_OFFSET[11:1]										DMA1_ACT	DMA2_ACT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-12. DMA Configuration register (DMA_CONFIG)

Table 23-13. DMA_CONFIG field descriptions

Field	Description
DMA1_CNT[11:0]	Byte count to be transferred by DMA for DMA channel 1.

Table 23-13. DMA_CONFIG field descriptions (continued)

Field	Description
DMA2_CNT[6:0]	Byte count to be transferred by DMA for DMA channel 2.
DMA2_OFFSET[11:1]	Byte offset for DMA channel 2. DMA channel 2 transfer start s at this offset count. DMA2_OFFSET must be 256 bytes aligned, so DMA2_OFFSET[7:1] should be always 0.
DMA1_ACT	DMA Channel 1 Activity 0 DMA channel 1 is inactive. 1 DMA channel 1 is active, and will be used for transfer to memory when triggered.
DMA2_ACT	DMA Channel 2 Activity 0 DMA channel 2 is inactive. 1 DMA channel 2 is active, and is used for transfer to memory when triggered.

23.7.11 Cache Swap register (CACHE_SWAP)

Address: Base + 0x3F28 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	CACHE_SWAP_ADDR2[13:3]											0
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	CACHE_SWAP_ADDR1[13:3]											0
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0

Figure 23-13. Cache Swap register (CACHE_SWAP)

Table 23-14. CACHE_SWAP field descriptions

Field	Description
CACHE_SWAP_ADDR1[13:3]	Lower swap address.
CACHE_SWAP_ADDR2[13:3]	Upper swap address.

Note: When DMA transfers data to/from the NFC cache(NFC SRAM buffer), or when CPU reads or writes data to/from the NFC cache via the IPS bus, all accesses that should go to CACHE_SWAP_ADDR1, are directed to CACHE_SWAP_ADDR2, and all accesses that should go to CACHE_SWAP_ADDR2 are directed to CACHE_SWAP_ADDR1.

The feature is added to the flash controller to allow to have the bad block marker in the first position of the spare area of a page. Because of the way the flash controller interleaves data and ECC bytes on flash devices with page sizes larger than 2 KB, the position of the bad block marker is shifted, and will not appear in the first position of the spare area of the page any more. The cache swap feature allows consistent swapping of the “actual” bad block line with the “expected” bad block line, and causes the operating system to get the bad block marker in the position where it is expected. Table 23-21 gives some examples of usage.

23.7.12 Sector Size register (SECTOR_SIZE)

Address: Base + 0x0x3F2C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	SECTOR_SIZE[12:0]												
W																
Reset	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0

Figure 23-14. Sector Size register (SECTOR_SIZE)

Table 23-15. SECTOR_SIZE field descriptions

Field	Description
SECTOR_SIZE[12:0]	Size in bytes of one elementary transfer unit. For devices with pages of 2 KB and smaller, this is the physical size of the page in bytes (data bytes + header bytes + ECC bytes) transferred in one page. When pages are larger than 2 KB, they need to be split in multiple virtual pages. In this case, the sector size is the size of the virtual page. The virtual page size is the physical size divided by the splitting factor PAGE_CNT. Table 23-20 gives examples on programming this field. **Note:** If only a part of a page to be programmed or read, SECTOR_SIZE can be set to the number of affected bytes, not the page size. Then ECC and DMA are all performed on the number of bytes, indicated by SECTOR_SIZE. **Note:** For 16-bit data width flash devices, only odd sector_size is supported. If sector_size is even number, the real implemented SECTOR_SIZE = SECTOR_SIZE – 1. So if SECTOR_SIZE = 1, there would be no data being written or read.

23.7.13 Flash Configuration register (FLASH_CONFIG)

Address: Base + 0x3F30 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	STOP_ON_WERR	ECC_SRAM_ADDR[11:3]									ECC_SRAM_REQ	DMA_REQ	ECC_MODE[2:0]			FAST_FLASH
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ID_COUNT[2:0]			CMD_TIMEOUT[4:0]					16BIT	BOOT_MODE	ADDR_AUTO_INCR	BUFNO_AUTO_INCR	PAGE_CNT[3:0]			
Reset	1	0	1	0	0	1	1	0	0	note	1	1	0	0	0	1

Figure 23-15. Flash Configuration register (FLASH_CONFIG)

Table 23-16. FLASH_CONFIG field descriptions

Field	Description
STOP_ON_WERR	Stop on Write Error. 0 No stop on write error. 1 Auto sequencer (see Table 23-22) will stop on write error.
ECC_SRAM_ADDR[11:3]	Byte address in SRAM where ECC status is written.
ECC_SRAM_REQ	0 Do not write ECC status to SRAM. 1 Write ECC status to SRAM.
DMA_REQ	0 Do not transfer sector after ECC done. 1 After ECC done, transfer sector using DMA.
ECC_MODE[2:0]	ECC mode. 000 No correction, ECC bypass. 001 4-error correction (8 ECC bytes). 010 6-error correction (12 ECC bytes). 011 8-error correction (15 ECC bytes). 100 12-error correction (23 ECC bytes). 101 16-error correction (30 ECC bytes). 110 24-error correction (45 ECC bytes). 111 32-error correction (60 ECC bytes).
FAST_FLASH	0 Slow flash timing. Clock in read data on rising edge of read strobe. 1 Fast flash timing. Clock in read data 1/2 clock later than rising edge of read strobe See Section 23.8.4, “Fast Flash Configuration for EDO,” for more information.
ID_COUNT[2:0]	Number of bytes that will be read for the *read id* command.
CMD_TIMEOUT[4:0]	The number of flash_clk cycles from $\overline{\text{NFC\_WE}}$ high to NAND flash busy (t_{WB}), or from $\overline{\text{NFC\_WE}}$ high to $\overline{\text{NFC\_RE}}$ low (tWHR). After last command is issued to flash, before sample NFC_R/$\overline{\text{B}}$, NFC must wait for t_{WB}. After t_{WB}, if NFC_R/$\overline{\text{B}}$ is sampled high, NFC considers the command to be timed out, and the flash is idle. NFC can issue new commands to the flash. If NFC_R/$\overline{\text{B}}$ is sampled low, the NAND flash is busy. When reading status or ID from the NAND flash, after the last command is issued to the flash, the NFC must wait for t_{WHR}, then assert $\overline{\text{NFC\_RE}}$ low, to read out valid status or ID. **Note:** t_{WB} exists in page program/read, block erase,etc. Refer to the NAND flash datasheet for details of t_{WB} and t_{WHR}.
16BIT	0 8-bit wide flash mode. 1 16-bit wide flash mode.

Table 23-16. FLASH_CONFIG field descriptions (continued)

Field	Description
BOOT_MODE	Boot mode. 0 Normal mode. 1 Boot mode. **Note:** Boot_mode power-on-reset value is 0 if no boot is performed from the NFC, it is 1 if boot is performed.
BUFNO_AUTO_INCR	0 Do not auto-increment buffer number. 1 Auto increment buffer number.
ADDR_AUTO_INCR	0 Do not auto-increment flash row address. 1 Auto-increment flash row address.
PAGE_CNT[3:0]	Number of virtual pages (in one physical flash page) to be programmed or read, etc.

23.7.14 DMA2 Address register (DMA2_ADDR)

Address: Base + 0x3F34 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DMA2_ADDR[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DMA2_ADDR[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-16. DMA2 Address register (DMA2_ADDR)

Table 23-17. DMA2_ADDR field descriptions

Field	Description
DMA2_ADDR[31:0]	DMA channel 2 address

23.7.15 IRQ and Status register (IRQ_STATUS)

Address: Base + 0x3F38 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WERR_IRQ	CMD_DONE_IRQ	IDLE_IRQ	0	WERR_STATUS	FLASH_CMD_BUSY	RESIDUE_BUSY	ECC_BUSY	DMA_BUSY	WERR_EN	CMD_DONE_EN	IDLE_EN	WERR_CLEAR	CMD_DONE_CLEAR	IDLE_CLEAR	0
W													w1c[1]	w1c[2]	w1c[3]	
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	RESIDUE_BUFF_NO		ECC_BUFF_NO		DMA_BUFF_NO	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-17. IRQ and Status register (IRQ_STATUS)

1 When written, this bit clears itself and the WERR_IRQ bit.

2 When written, this bit clears itself and the CMD_DONE_IRQ bit.

3 When written, this bit clears itself and the IDLE_IRQ bit.

Table 23-18. IRQ_STATUS field descriptions

Field	Description
WERR_IRQ	Write error interrupt. Set if an error condition is detected during a “flash read status” command. This bit is cleared when the WERR_CLEAR bit is written.
CMD_DONE_IRQ	Command done interrupt. Set if command processing is done. This bit is cleared when the CMD_DONE_CLEAR bit is written.
IDLE_IRQ	Command idle interrupt. Set if command done, residue engine, ECC engine, and DMA engine are idle. This bit is cleared when the IDLE_CLEAR bit is written.
WERR_STATUS	Write error status. High if an eror condition was detected during the last “flash read status” command. This bit is cleared when the WERR_CLEAR bit is written.
FLASH_CMD_BUSY	Set if command execution is busy, cleared otherwise.
RESIDUE_BUSY	Set if the residue engine is busy, cleared otherwise.
ECC_BUSY	Set if the ECC engine is busy, cleared otherwise.
DMA_BUSY	Set if the DMA engine is busy, cleared otherwise.
WERR_EN	Enable bit for WERR_IRQ. 0 Write error interrupt is disabled. 1 Write error interrupt is enabled.
CMD_DONE_EN	Enable bit for CMD_DONE_IRQ. 0 Command done interrupt is disabled. 1 Command done interrupt is enabled.
IDLE_EN	Enable bit for command IDLE_IRQ. 0 Command idle interrupt is disabled. 1 Command idle interrupt is enabled.
WERR_CLEAR	Clear bit for WERR_IRQ. Writing ‘1’ to this bit clears WERR_IRQ.

Table 23-18. IRQ_STATUS field descriptions (continued)

Field	Description
CMD_DONE_CLEAR	Clear bit for CMD_DONE_IRQ. Writing ‘1’ to this bit clears CMD_DONE_IRQ.
IDLE_CLEAR	Clear bit for IDLE_IRQ. Writing ‘1’ to this bit will clear IDLE_IRQ.
RESIDUE_BUFF_NO	Residue buffer number. Buffer number corresponding with the current residue block task.
ECC_BUFF_NO	ECC buffer number. Buffer number corresponding with the current ECC task.
DMA_BUFF_NO	DMA buffer number. Buffer number corresponding with the current DMA task.

There are two interrupts to flag the end of a command execution:

- The *done* interrupt CMD_DONE_IRQ
- The *cidle (command idle)* interrupt IDLE_IRQ

Use the CMD_DONE_IRQ interrupt if commands are sent back-to-back to the flash. It gives an indication when a new command can be dispatched. The *done* interrupt is given *before* the flash data is corrected and resident in memory, because operation of the ECC engine and DMA engine is pipelined.

When the CMD_DONE_IRQ interrupt is used to track command completion, the software may also monitor the ECC_BUSY, DMA_BUSY, ECC_BUFF_NO, and DMA_BUFF_NO bitfields. The ECC_BUSY bitfield indicates that the ECC block is still busy, and gives the number of the buffer on which the ECC block is working in the ECC_BUFF_NO bitfield. The DMA_BUSY bitfield indicates that the DMA block is still busy, and gives the number of the the buffer on which the DMA block is working in the DMA_BUFF_NO bitfuield.

Use the IDLE_IRQ interrupt if you want to use the data produced in the next process. The IDLE_IRQ interrupt indicates that all command processing has terminated, and the relevant data is now available either in memory or in the NFC SRAM buffer. When using back-to-back reads to the flash, use of the IDLE_IRQ interrupt means the NFC does not operate at its maximum transfer speed, as ECC and DMA are now done in foreground.

When using the CMD_DONE_IRQ interrupt, transfer completion for write pages can be assumed when the CMD_DONE_IRQ interrupt is received. When CMD_DONE_IRQ is received for read pages, the data may still be in flight in the DMA or the ECC. To check this, the CPU should remember the buffer number (*xxx*_BUFNO bitfield) associated with the command, and wait until the DMA and ECC are either idle, or are both busy on a different buffer number. (ECC buffer number and DMA buffer number fields do not match *xxx*_BUFNO specified with command.) Checking can be done on any CMD_DONE_IRQ interrupt, or by polling the register.

23.8 Functional Description

The NFC executes commands on an external, or bank of external NAND Flash chips. The commands supported include read, program, reset, erase, status read, read ID, etc.

The NFC block has DMA engine and built-in ECC logic. For each read or write, ECC calculations are performed on-the-fly. Two DMA channels are organized for each read or write: One for the main area, and one for the spare area. It is possible to disable the second DMA channel, and transfer main and spare data with just the first DMA channel.

Page size supported is 512 bytes, 2 KB, 4 KB, and 8 KB. Eight different ECC settings are provided: 0-, 4-, 6-, 8-, 12-, 16-, 24-, and 32-bit errors. They use 0, 8, 12, 15, 23, 30, 45, or 60 ECC bytes. ECC works on page sizes of 512 + spares bytes, 1 KB + spares bytes, and 2 KB + spares bytes. The ECC algorithm used is a BCH code.

ECC is performed on the fly, during both read and write.

The error corrector can write ECC status to the spare area. This is done because the read is pipelined. This means, while current page is being transferred from flash to buffer, previous page is ECC corrected, and the page before that is being transferred using DMA. The following method has been devised to inform the CPU of ECC errors. The ECC status is written to the aux area of the sector and transferred to memory. See Section 23.8.1, “Error Corrector Status,” for more information. The CPU must inspect the ECC result in memory, and act appropriately.

Reads are pipelined, as already described. Writes are not pipelined. Write is flow-through, and no advance operations are done during write. If a problem is found during write, the command sequence may be interrupted, and the CPU is informed.

Each page read, page write, page erase, read ID, or read status command sequence needs CPU attention only once. The CPU needs to prepare the DMA to point to the data, write correct values to all registers, and start the command. After command completion, the NFC may interrupt the CPU.

The NFC allows command repeat. Command repeat is useful for writes, reads, and erases, and allows processing multiple pages with just one command given by the CPU. No bank interleaving is supported during command repeat.

After power-on reset, a reset command (0xFF) is sent to the flash.

Boot from NAND flash is optional. The feature is activated when *boot_after_reset* input is high during power-on reset. If boot feature is activated, the NFC will read 4 pages from block 0. Each page is 1056 bytes. The boot pages are protected by 32-bit error correction, which means that of the 1056 bytes, 996 bytes are user bytes, and 60 bytes are ECC bytes. When the data from the boot pages is read, successfully error corrected, and stored in the NFC SRAM, the flag *boot_done* is pulled high, indicating to the CPU that its boot code is visible in the NFC SRAM, and visible on addresses 0 to 3983 (decimal) (3984 bytes total).

In case the boot image from sector 0 cannot be corrected, because there are more than 32 errors in one or several pages, boot is retried on the blocks at row addresses 256, 512, and 768. If it still fails after these retries, the *boot_fail* flag is pulled high, and boot is given up.

Right after boot, a special address hashing function is active on all reads and writes done to NFC SRAM. This hashing function interleaves the page data from the 4 boot pages in such a way that all user data is visible in address range 0 to 3983 instead of four different ranges, each for one page. This hashing is controlled by the FLASH_CONFIG[BOOT_MODE] bit, and the hashing should be turned off by the CPU after finishing reading/executing the boot image, and before normal operations of the NFC. Reference Figure 23-25 and Section 23.5, “NFC Buffer Memory Space.”

Page size at boot is set to 1056 bytes, to be compatible with a large number of NFC devices, without needing additional power-on reset flags to indicate boot device.

- Compatible with 8-wide SLC and MLC devices with page size of 2048 bytes + 64 bytes spare

- Compatible with 8-wide SLC and MLC devices with page size of 4 KB and larger
- Compatible with 16-wide SLC and MLC devices with page size of 2048 bytes + 64 bytes spare
- Compatible with 16-wide SLC and MLC devices with page size of 4 KB and larger
- Not compatible with devices with 512 bytes page size.

23.8.1 Error Corrector Status

The ECC engine determines if a page is correctable. If correctable, it corrects error bits, and indicates error number, otherwise, the Corfail bit is asserted as shown in Table 23-19. In order for a bad block management strategy to work, it may be necessary for the processor to obtain this information.

The error corrector writes the status word to a byte location to the SRAM buffer, defined by ECC_SRAM_ADDR + 0x07. It is selectable if the status is written or not with bit ECC_SRAM_REQ. Both fields are part of config register FLASH_CONFIG. In case the status is written to the SRAM buffer, it becomes effectively part of the flash data, and will be processed like the flash data. Most likely, the status byte will be written to memory as part of the page header. Once in memory, the ECC status is visible to the CPU, while CPU parsing the rest of the flash header too. No interrupt on error or status is available because this would increase the interrupt load on the CPU. (The interrupt would be independent of the CMD_DONE_IRQ interrupt.) It is not possbile to stop reading when ECC failed.

The organization of the status byte is given in Table 23-19.

Table 23-19. ECC status word

Bitfield	Name	Meaning
7	Corfail	0 Page has been successfully corrected. 1 Page is uncorrectable.
5:0	error_count	Number of errors that have been corrected in this page.

23.8.2 NFC Basic Commands

23.8.2.1 Page Read

This command is used to read pages from the NAND flash. Figure 23-18 is the general flow chart of the read operation.

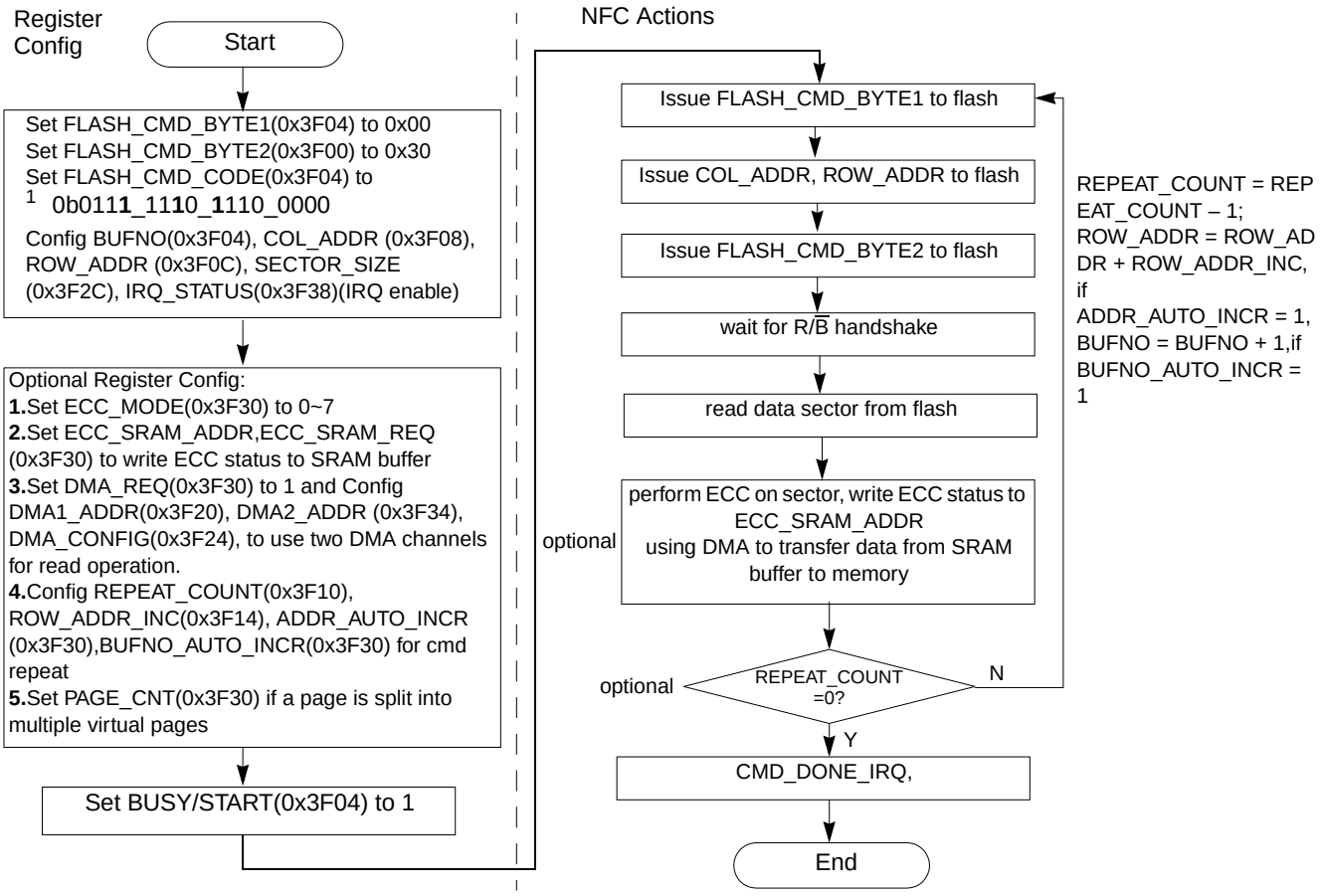


Figure 23-18. Flow Chart of Read Operation

1 COL_ADDR2, ROW_ADDR3,FLASH_CMD_BYTE2(bolded) are not necessary for some flash, see datasheets for detail. For example, if flash only has 1 column address, then FLASH_CMD_CODE = 0b011**0**_1110_1110_0000; if flash only has 2 row addresses, then FLASH_CMD_CODE = 0b0111_11**0**0_1110_0000; if flash does not need the second command 30h for read, then FLASH_CMD_CODE = 0b0110_1110_**0**110_0000.

Figure 23-19 shows a particular case: one page is split into 8 virtual pages (see Section 23.8.5, “Organization of Data in the NAND Flash”), and DMA is not used. The SRAM buffer can hold data for 4 (virtual) pages at most, CPU must transfer data out of SRAM buffer after the first 4 virtual pages are read from flash, otherwise, the next 4 virtual pages data will overwrite the buffer. So read operation has following steps:

1. Configure registers as Figure 23-18, PAGE_CNT = 4, start commands, wait for CMD_DONE_IRQ.
2. CPU reads data from buffer, set FLASH_CMD_CODE = 0b10_0000 (only enable read data).
3. Start commands to read out the next 4 virtual pages, wait for CMD_DONE_IRQ.

If DMA is used to transfer data from SRAM buffer to memory instead of CPU, the flow in Figure 23-18 is used: set PAGE_CNT = 8, set DMA_REQ = 1, configure DMA registers, start commands. A pipeline (see Section 23.8, “Functional Description”) controls the read operation.

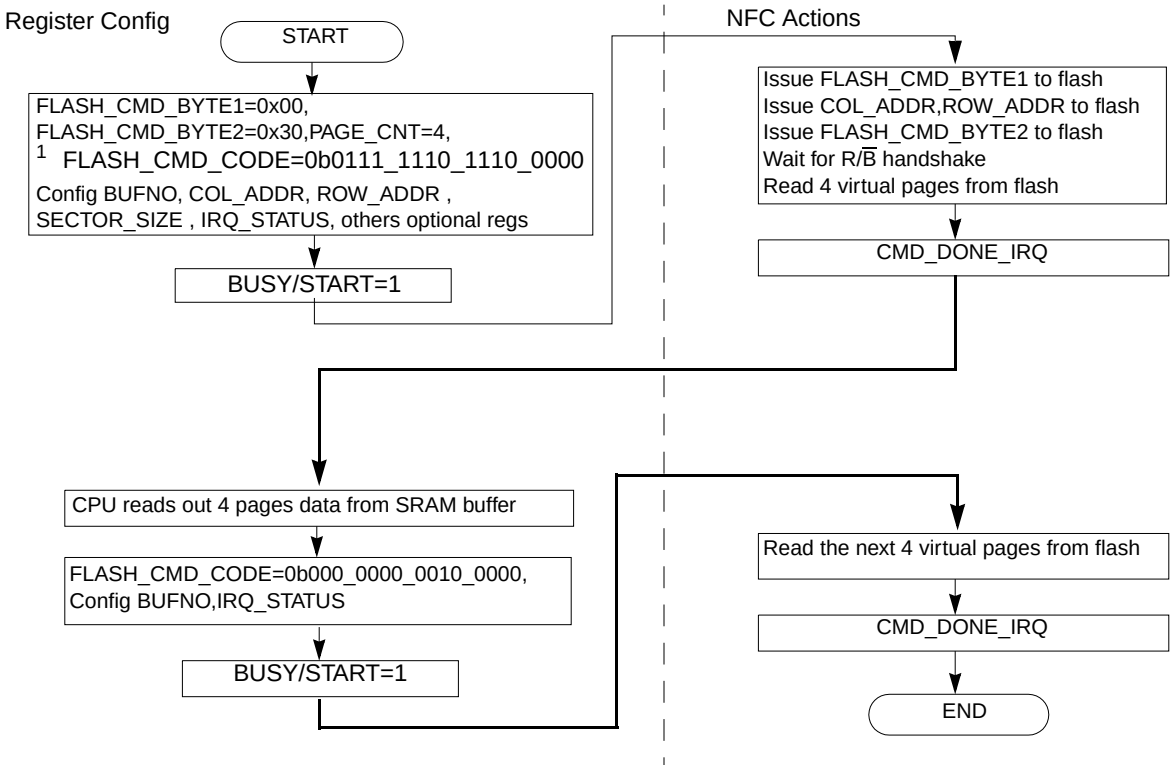


Figure 23-19. Flow Chart of Read Operation, PAGE_CNT = 8, No DMA

1 Reference the note in Figure 23-18.

23.8.2.2 Page Program

This command is used to program pages to the NAND Flash. Figure 23-20 is the general flow of page program operation.

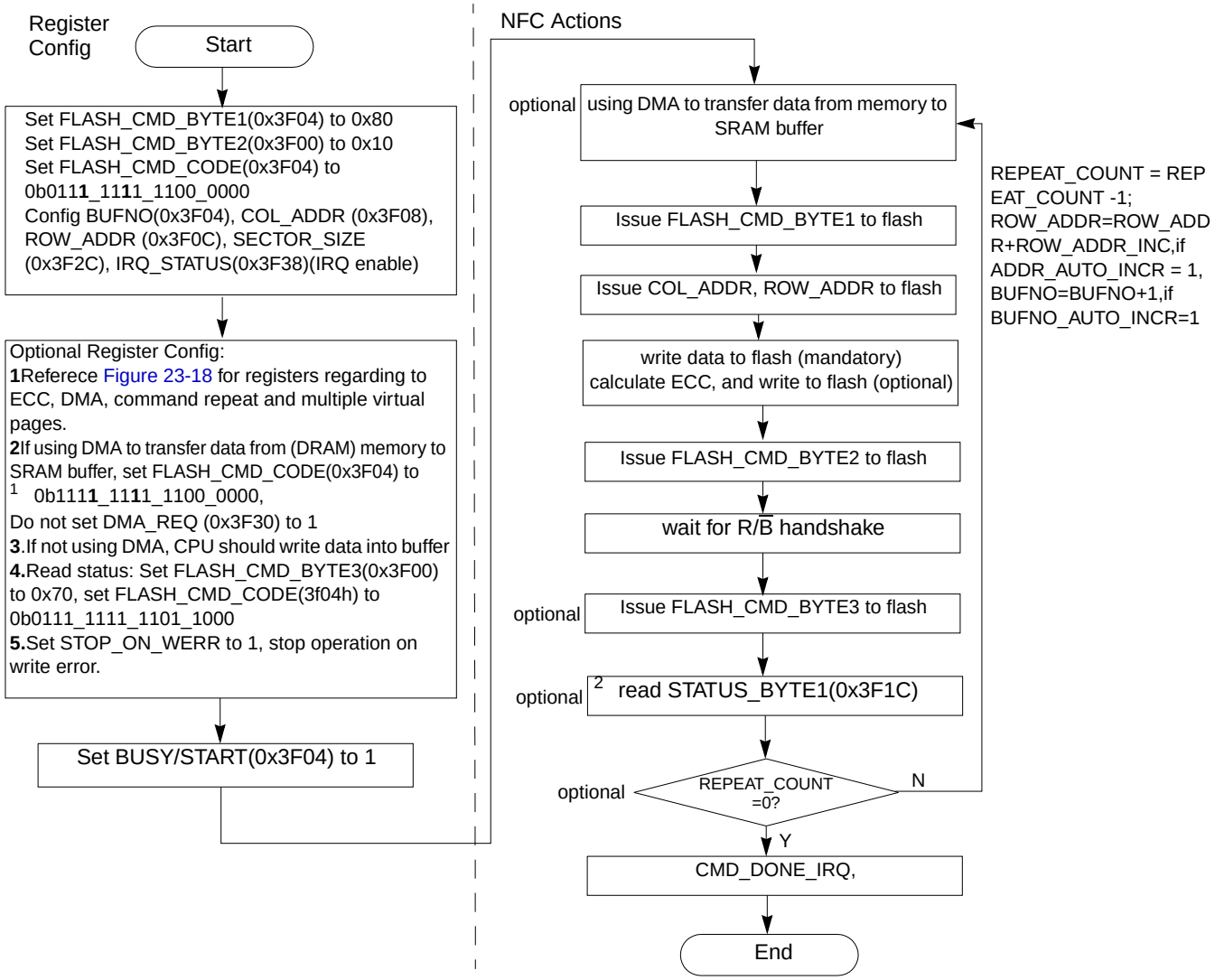


Figure 23-20. Flow Chart of Page Program Operation

1 COL_ADDR2, ROW_ADDR3 (bolded) are not necessary for some flash, see datasheets for detail. Reference the footnote of Figure 23-18.

2 If STOP_ON_WERR = 1 and STATUS_BYTE1[0] = 1, the operation stops.

Figure 23-21 is the particular case which is similar as Figure 23-19. CPU writes 4 virtual pages data into buffer at most before the first time start commands. PAGE_CNT = 4, FLASH_CMD_CODE is set twice:

- First time 0b0111_1111_0000_0000, NFC issues FLASH_CMD_BYTE1(0x80), address cycles, 4 virtual pages data to flash. After CMD_DONE_IRQ, CPU can write the next 4 virtual pages data into SRAM buffer;
- Second time 0b0000_0001_1100_0000, NFC sends the next 4 virtual pages data to flash, issue FLASH_CMD_BYTE2(0x10), wait for R/$\overline{B}$ handshake, then wait for the CMD_DONE_IRQ.

Like read operation, if DMA is used instead of CPU to transfer data from memory to NFC SRAM buffer, the flow in Figure 23-20 is used, PAGE_CNT = 8.

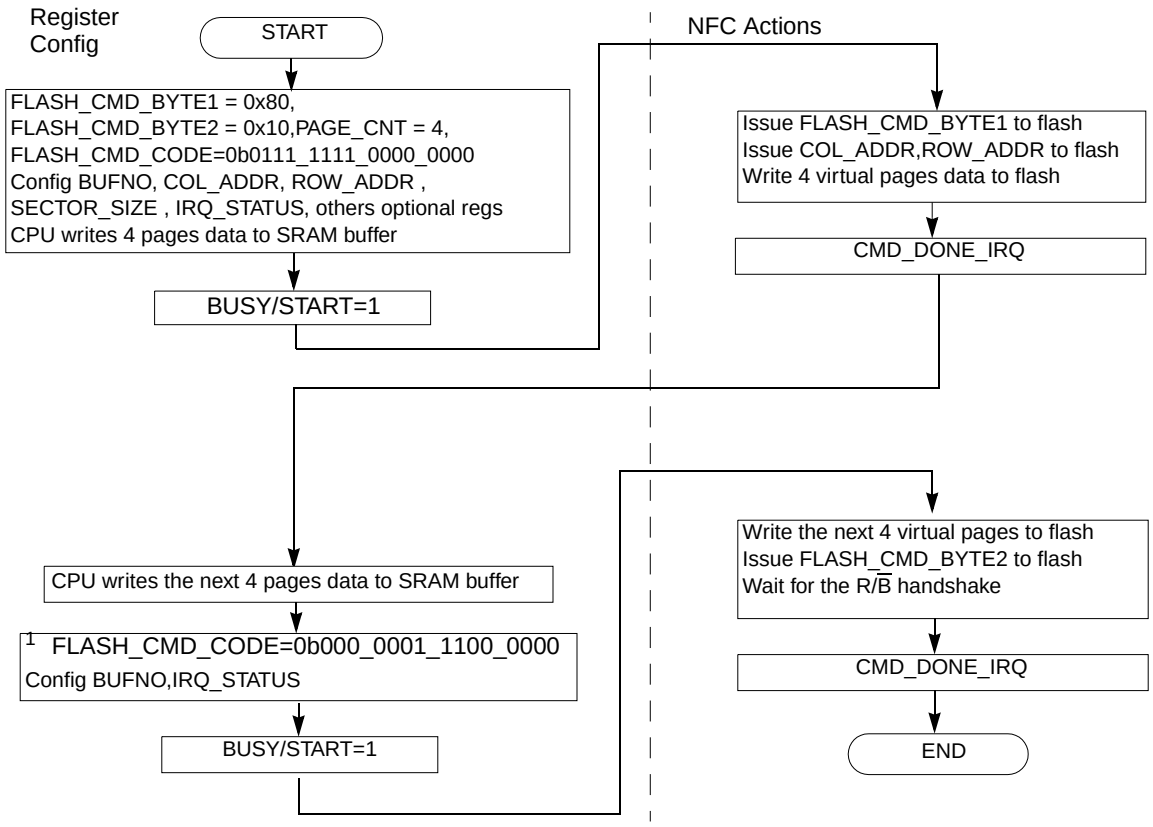


Figure 23-21. Flow Chart of Page Program Operation, PAGE_CNT = 8, No DMA

1 If want to read status after second command 0x10 issued, set FLASH_CMD_BYTE3 = 0x70, FLASH_CMD_CODE = 0b0000_0001_1101_1000. Then after "Wait for the R/B handshake", NFC issues FLASH_CMD_BYTE3 to flash, and read status. If STOP_ON_WERR = 1 and STATUS_BYTE1[0] = 1, operation stops, otherwise, CMD_DONE_IRQ comes out. The COL_ADDR2 and ROW_ADDR3 of the first FLASH_CMD_CODE may not be necessary, reference note 1 of Figure 23-18.

23.8.2.3 Block Erase

This command is used to erase blocks.

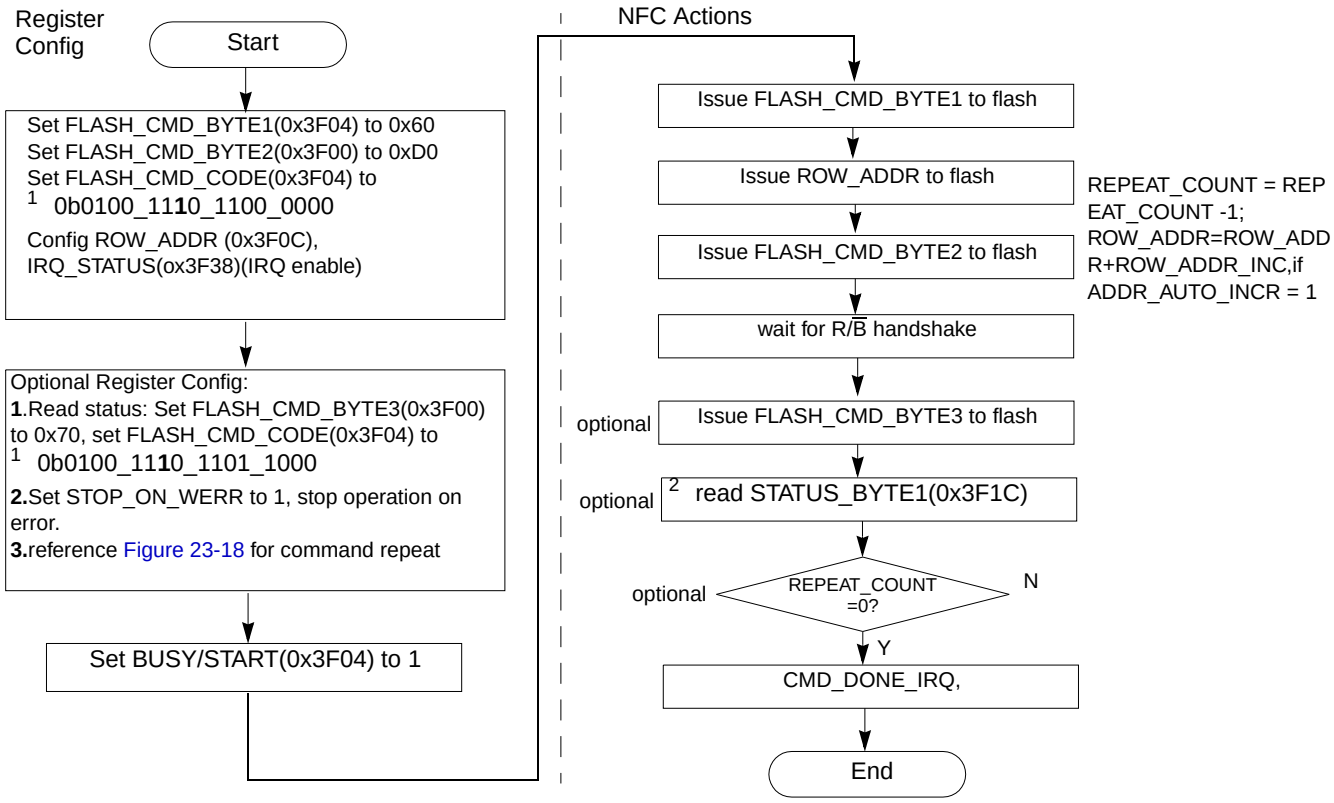


Figure 23-22. Flow Chart of Block Erase Operation

1 ROW_ADDR3(bolded) is not necessary for some flashes, see datasheets for detail.
2 If STOP_ON_WERR = 1 and STATUS_BYTE1[0] = 1, the operation stops.

23.8.2.4 Read ID

This command is used to read the flash ID.

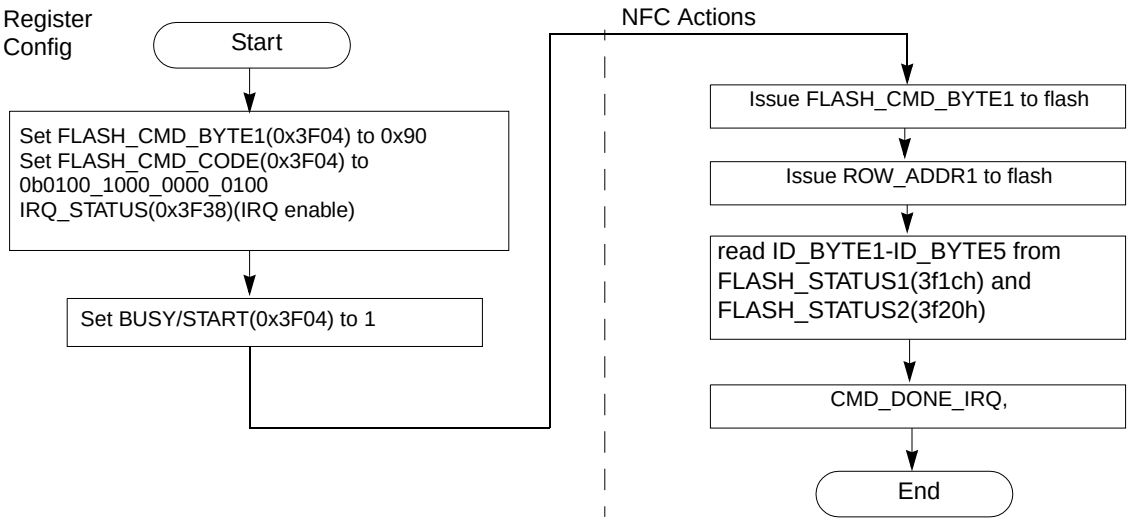


Figure 23-23. Flow Chart of Read ID Operation

23.8.2.5 Reset

This command is used to sent a single reset command to the flash.

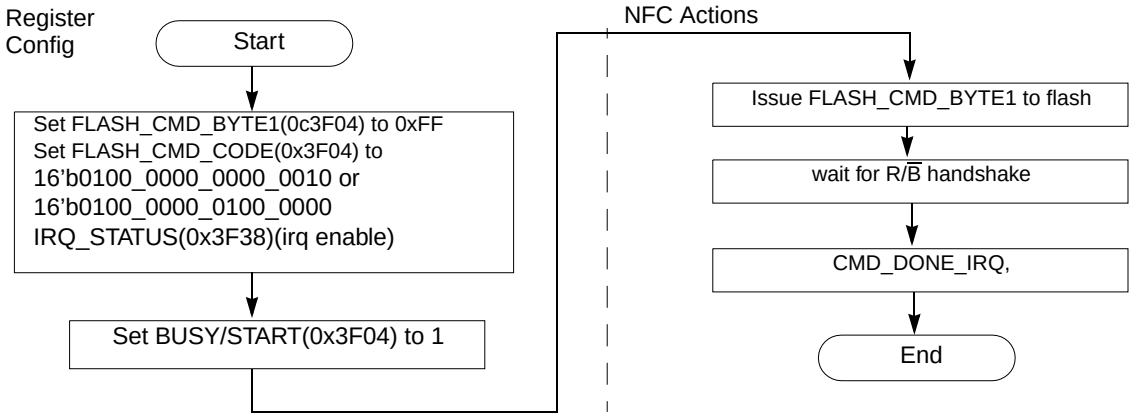


Figure 23-24. Flow Chart of Reset Operation

23.8.3 NAND Flash boot

Power-on reset values of some registers (for boot) are:

- Sector size is 1056 bytes: SECTOR_SIZE = 1056

- Flash is defined as 8-bit: 16BIT = 0
- ECC correction depth is 32 bits errors: ECC_MODE = 0b111
- Boot sector address is 0: BUFNO = 0b00

Boot-up occurs after power-on reset. After boot, the following happens.

1. 4 boot blocks are identified in the flash. These are blocks at row address 0, 256, 512, and 768.
2. The flash controller does a burst read from the first boot block to memory.The burst is 4 pages long, so a total of 4 KB are read in this way.
3. If no ECC failure occurs during the burst read, boot is successful.
4. If an ECC failure occurs during the burst read, the process is started again from the next boot block. If the fourth block still has an ECC failure, the boot is unsuccessful, , which means it is not possible to read a reliable boot image from the flash.
5. CPU access is held until boot completion.
6. After boot, boot image is visible on IPS memory map, address 0 to 3983. A special hash function is active on the NFC SRAM read to have the boot image in one continuous address range, and not in 4 address ranges, one for each page. The config bit BOOT_MODE controls this hash function, and this bit should be turned off after the CPU has read/executed the boot image, and before it operates the NFC in the standard mode. See Figure 23-25.

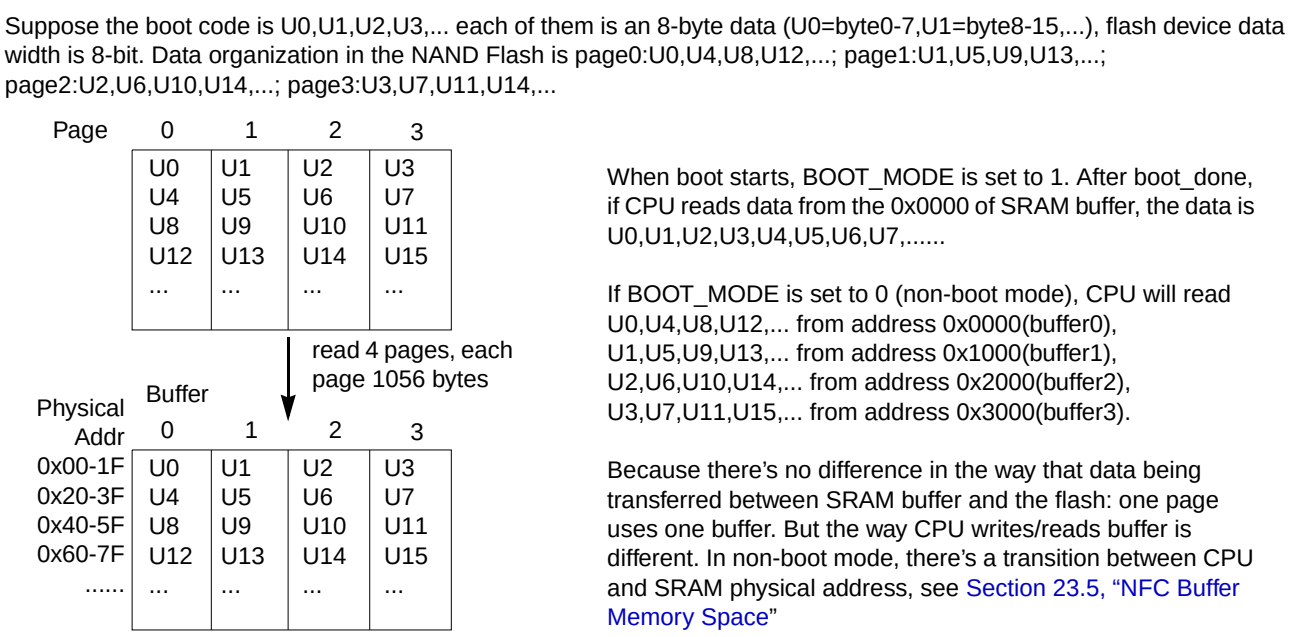
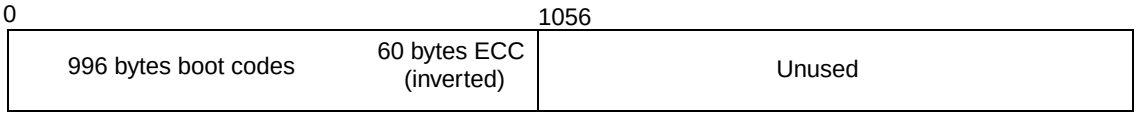


Figure 23-25. Boot and BOOT_MODE

The boot is identical for a 8-wide and 16-wide flash. If a 16-wide flash is connected, the data on the upper lane are discarded. If a 8-wide flash is connected, the upper 1 KB of data are not read. See Figure 23-26.

8-bit flash boot codes and ECC's location in a page



16-bit flash boot codes and ECC's location in a page

	[15:8]	[7:0]
0x0000	XX	boot code byte 0
0x0002	XX	boot code byte 1
0x0004	... XX ...	... boot code byte i ...
.....		
0x07C6	XX	boot code byte 995
0x07C8	XX	ECC byte 0 (inverted)
0x07CA	XX	ECC byte 1 (inverted)
0x07CC	... XX ...	... ECC byte j (inverted) ...
.....		
0x083E	XX	ECC byte 59 (inverted)

Figure 23-26. Boot data's location in a page of different size flash

23.8.4 Fast Flash Configuration for EDO

Normally, read out data goes valid after the H → L transition of $\overline{RE}$, and invalid on the L → H transition $t_{RHOH} < t_{REH}$.

Some flash devices have the EDO (enhanced data out) feature, the data can be held until the next $\overline{RE}$ H → L transition, $t_{RHOH} > t_{REH}$.

To support the EDO feature, NFC must work in fast mode: set FAST_FLASH = 1, Section 23.7.13, "Flash Configuration register (FLASH_CONFIG)." The flash_clk should be configured fast enough (usually > 33 MHz) according to the data sheet of flash devices.

23.8.5 Organization of Data in the NAND Flash

Pages on the flash can be split into multiple ECC/DMA pages. The parameter that controls this is PAGE_CNT[3:0] in the FLASH_CONFIG register. This parameter gives the number of virtual, ECC/DMA pages in one flash page. See Section 23.7.13, "Flash Configuration register (FLASH_CONFIG)," for more details.

The virtual page itself is split in *user area* and *ECC area*. Data in the user area can be set or used by the application. Data in the ECC area is set and used by the ECC.

The following tables give virtual-to-physical mappings for various flash devices and their recommended settings.

Table 23-20. [1]Virtual-to-physical mappings of different flash[2]

Flash page size (main + spare) bytes	ECC_ MODE	ECC bits	PAGE_ CNT	Sector size bytes	virtual page user size bytes	Mapping
512 + 16	000	0	1	528	528	VirtualPage_0[527:0] = Physical[527:0]
512 + 16	001	4	1	528	520	VirtualPage_0[519:0] = Physical[519:0]
2048 + 64	000	0	1	2112	2112	VirtualPage_0[2111:0] = Physical[2111:0]
2048 + 64	101	16	1	2112	2082	VirtualPage_0[2081:0] = Physical[2081:0]
2048 + 64	110	24	1	2112	2067	VirtualPage_0[2066:0] = Physical[2066:0]
2048 + 64	111	32	1	2112	2052	VirtualPage_0[2051:0] = Physical[2051:0]
2048 + 64	000	0	4	528	528	VirtualPage_0[527:0] = Physical[527:0] VirtualPage_1[527:0] = Physical[1055:528] VirtualPage_2[527:0] = Physical[1583:1056] VirtualPage_3[527:0] = Physical[2111:1584]
2048 + 64	001	4	4	528	520	VirtualPage_0[519:0] = Physical[519:0] VirtualPage_1[519:0] = Physical[1047:528] VirtualPage_2[519:0] = Physical[1575:1056] VirtualPage_3[519:0] = Physical[2103:1584]
4096 + 128	000	0	2	2112	2112	VirtualPage_0[2111:0] = Physical[2111:0] VirtualPage_1[2111:0] = Physical[4223:2112]
4096 + 128	101	16	2	2112	2082	VirtualPage_0[2081:0] = Physical[2081:0] VirtualPage_1[2081:0] = Physical[4193:2112]
4096 + 128	110	24	2	2112	2067	VirtualPage_0[2066:0] = Physical[2066:0] VirtualPage_1[2066:0] = Physical[4178:2112]
4096 + 128	111	32	2	2112	2052	VirtualPage_0[2051:0] = Physical[2051:0][3] VirtualPage_1[2051:0] = Physical[4163:2112]
4096 + 128	000	0	[4]8	528	528	VirtualPage_0[527:0] = Physical[527:0] VirtualPage_1[527:0] = Physical[1055:528] VirtualPage_2[527:0] = Physical[1583:1056] VirtualPage_3[527:0] = Physical[2111:1584] VirtualPage_4[527:0] = Physical[2639:2112] VirtualPage_5[527:0] = Physical[3167:2640] VirtualPage_6[527:0] = Physical[3695:3168] VirtualPage_7[527:0] = Physical[4223:3696]
4096 + 128	001	4	8	528	520	VirtualPage_0[519:0] = Physical[519:0] VirtualPage_1[519:0] = Physical[1047:528] VirtualPage_2[519:0] = Physical[1575:1056] VirtualPage_3[519:0] = Physical[2103:1584] VirtualPage_4[519:0] = Physical[2631:2112] VirtualPage_5[519:0] = Physical[3159:2640] VirtualPage_6[519:0] = Physical[3687:3168] VirtualPage_7[519:0] = Physical[4215:3696]
4096 + 208	000	0	2	2152	2152	VirtualPage_0[2151:0] = Physical[2151:0] VirtualPage_1[2151:0] = Physical4303:2152]

Table 23-20. [1]Virtual-to-physical mappings of different flash[2] (continued)

Flash page size (main + spare) bytes	ECC_ MODE	ECC bits	PAGE_ CNT	Sector size bytes	virtual page user size bytes	Mapping
4096 + 208	101	16	2	2152	2122	VirtualPage_0[2121:0] = Physical[2121:0] VirtualPage_1[2121:0] = Physical[4273:2152]
4096 + 208	110	24	2	2152	2104	VirtualPage_0[2103:0] = Physical[2103:0] VirtualPage_1[2103:0] = Physical[4255:2152]
4096 + 208	111	32	2	2152	2092	VirtualPage_0[2091:0] = Physical[2091:0] VirtualPage_1[2091:0] = Physical[4243:2152]
4096 + 208	000	0	8	538	538	VirtualPage_0[537:0] = Physical[537:0] VirtualPage_1[537:0] = Physical[1075:538] VirtualPage_2[537:0] = Physical[1613:1076] VirtualPage_3[537:0] = Physical[2151:1614] VirtualPage_4[537:0] = Physical[2689:2152] VirtualPage_5[537:0] = Physical[3227:2690] VirtualPage_6[537:0] = Physical[3765:3228] VirtualPage_7[537:0] = Physical[4304:3766]
4096 + 208	001	4	8	538	530	VirtualPage_0[529:0] = Physical[529:0] VirtualPage_1[529:0] = Physical[1067:538] VirtualPage_2[529:0] = Physical[1605:1076] VirtualPage_3[529:0] = Physical[2143:1614] VirtualPage_4[529:0] = Physical[2681:2152] VirtualPage_5[529:0] = Physical[3219:2690] VirtualPage_6[529:0] = Physical[3757:3228] VirtualPage_7[529:0] = Physical[4295:3766]
4096 + 208	010	6	8	538	526	VirtualPage_0[525:0] = Physical[525:0] VirtualPage_1[525:0] = Physical[1063:538] VirtualPage_2[525:0] = Physical[1601:1076] VirtualPage_3[525:0] = Physical[2139:1614] VirtualPage_4[525:0] = Physical[2677:2152] VirtualPage_5[525:0] = Physical[3215:2690] VirtualPage_6[525:0] = Physical[3753:3228] VirtualPage_7[525:0] = Physical[4291:3766]
4096 + 208	011	8	8	538	523	VirtualPage_0[522:0] = Physical[523:0] VirtualPage_1[522:0] = Physical[1060:538] VirtualPage_2[522:0] = Physical[1598:1076] VirtualPage_3[522:0] = Physical[2136:1614] VirtualPage_4[522:0] = Physical[2674:2152] VirtualPage_5[522:0] = Physical[3212:2690] VirtualPage_6[522:0] = Physical[3750:3228] VirtualPage_7[522:0] = Physical[4288:3766]

1 Not all settings shown. Its also possible to spit 2 KB and 4 KB sectors in 1 KB units, and this is not shown. For flash devices with more spare bytes, it is possible to correct more errors, as a higher ECC byte count cost.

2 It is possible to split 2 KB and 4 KB pages in virtual pages of 512, 1 KB or 2 KB. The best error protection is achieved with the larger page sizes. For example, splitting a 4 KB page into two 2 KB pages, with 32-bit error correction yields lower uncorrectable error probability than splitting the same page into eight 512-byte pages with 8-bit error correction. For OS compatibility, it may be necessary to use 512-byte pages, however.

3 In most applications, this mode is of no use because user size is too small.

[4] When a 4 KB page is split to eight virtual pages, if page program/read using DMA, set PAGE_CNT = 8, if not using DMA, set PAGE_CNT = 4. See Section 23.8.2.1, “Page Read,” and Section 23.8.2.2, “Page Program,” for more details.

In case flash devices are used with a physical page size of 4 KB or more, the bad block marker will appear as the first byte of the spare area, but because of the physical-to-virtual mapping, it will not appear in byte 2048 of the virtual page, where its logical place would be. The DMA engine contains the option to swap some bytes, and in order to make the bad block marker appear in the requested place.

Table 23-21. Use of swap field to get bad block marker in position 2048 with large page flash

Flash sector size (main + spare) bytes	Bad block marker (physical)	Bad block marker (virtual) before swap	Bad block marker (expected) After swap	Swap
4096 + 128	4096	Page 1/byte 1984	Page 1/byte 2048	CACHE_SWAP_ADDR1[13:3] = (1984/8)[1] CACHE_SWAP_ADDR2[13:3] = (2048/8)
4096 + 208	4096	Page 1/byte 1944	Page 1/byte 2048	CACHE_SWAP_ADDR1[13:3] = (1944/8) CACHE_SWAP_ADDR2[13:3] = (2048/8)

[1] Only works with a user page size of at least 2055 bytes. Does not work with ECC mode 111.

23.8.6 Flash Command Sequencer

There is a sequencer inside cmd_block. Sequencer state variable is *flash_req_phase[15:0]*, which is preloaded from FLASH_CMD_CODE. Each bit of *flash_req_phase* stands for a certain action. If the bit is set, the action is executed. If the bit is cleared, the action is not executed. *flash_req_phase* is preloaded from FLASH_CMD_CODE. When command repeat is enabled, the preload takes place for every tick of repeat count. There is additional code to control the reloading.

Table 23-22. Detail of flash_req_phase

Flash_req_ phase bit	Action when bit is set
15	Start DMA transfer to read data from memory , and write to SRAM.
14	Sent Command byte 1 to flash. See FLASH_CMD_BYTE1, Section 23.7.2, “Flash Command 2 register (FLASH_CMD2).”
13	Sent column address 1 to flash. See COL_ADDR1, Section 23.7.3, “Column Address register (COL_ADDR).”
12	Sent column address 2 to flash. See COL_ADDR2, Section 23.7.3, “Column Address register (COL_ADDR).”
11	Sent row address 1 to flash. See ROW_ADDR1, Section 23.7.4, “Row Address register (ROW_ADDR).”
10	Sent row address 2 to flash. See ROW_ADDR2, Section 23.7.4, “Row Address register (ROW_ADDR).”
9	Sent row address 3 to flash. See ROW_ADDR3, Section 23.7.4, “Row Address register (ROW_ADDR).”
8	Write data to flash. Total of PAGE_CNT (see Section 23.7.13, “Flash Configuration register (FLASH_CONFIG)”) pages is written to the flash, and equal number of starts is sent to the residue engine. Also, additional starts to the DMA engine are sent, until DMA has transferred all the PAGE_CNT data from memory to NFC.
7	Sent Command byte 2 to flash. See FLASH_CMD_BYTE2, Section 23.7.1, “Flash Command 1 register (FLASH_CMD1).”
6	Wait for flash R/$\overline{B}$ handshake.

Table 23-22. Detail of flash_req_phase (continued)

Flash_req_phase bit	Action when bit is set
5	Read data from flash. Read is only started if the new BUFNO is idle. See Section 23.7.2, “Flash Command 2 register (FLASH_CMD2).” One or more starts are sent to the residue engine, total PAGE_CNT starts. See Section 23.7.13, “Flash Configuration register (FLASH_CONFIG).” **Note:** For reads, DMA is not started. Instead, to start DMA for reads, DMA_REQ needs to be configured to 1. See Section 23.7.13, “Flash Configuration register (FLASH_CONFIG).”
4	Sent Command byte 3 to flash (see FLASH_CMD_BYTE3, Section 23.7.1, “Flash Command 1 register (FLASH_CMD1)”).
3	Read flash status.
2	Read ID.
1	Always set. End-of-command marker, used to signal “done.”

This page is intentionally left blank.

Chapter 24
Power Management Control Module (PMC)

24.1 Introduction

This chapter describes the power management control (PMC) module. The PMC is responsible for entering and exiting the low-power mode.

24.1.1 Features

The PMC module includes the following features:

- Causes the e300 Power Architecture core to enter low-power mode (Nap or Sleep) after the coherent system bus (CSB) is idle and DRAM controller is idle
- Causes the DRAM controller to enter and exit low-power (self-refresh) mode
- Optional interrupt when exiting low-power mode (Nap or Sleep)
- Exit low-power mode when the Power Architecture core is ready
- Enter and exit Deep Sleep mode (system OSC, system PLL, and e300 core PLL are put in low-power standby mode)
- Support e300 core PLL change and system clock divide ratio (SYS_DIV) copy mode

24.2 Memory Map and Register Definition

24.2.1 Memory Map

Table 24-1 shows the register memory map for the PMC module.

Table 24-1. PMC memory map

Offset from PMC_BASE (0xFF40_1000)[1]	Register	Access	Reset Value	Section/Page
0x00	PMC Configuration Register (PMC_PMCCR)	R/W	0x0000_0000	24.2.2.1/24-646
0x04	PMC Event Register (PMC_PMCER)	R/W	0x0000_0000	24.2.2.2/24-648
0x08	PMC Mask Register (PMC_PMCMR)	R/W	0x0000_0000	24.2.2.3/24-649
0x0C	PMC CORE_PLL Shadow Register (PMC_PMCSR)	R/W	0x0000_0000	24.2.2.4/24-649

Table 24-1. PMC memory map (Continued)

Offset from PMC_BASE (0xFF40_1000)[1]	Register	Access	Reset Value	Section/Page
0x10	PMC Wakeup Source Enable Register (PMC_PMCWSE)	R/W	0x0000_0000	24.2.2.5/24-650
0x14	PMC Wakeup Source Polarity Register (PMC_PMCWSP)	R/W	0x0000_00FF	24.2.2.6/24-651
0x18–0xFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

24.2.2 Register Descriptions

24.2.2.1 PMC Configuration Register (PMC_PMCCR)

Address: Base + 0x00 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	PRE DIV	CCM	DSM	DDR OFF	CORE OFF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24-1. PMC Configuration Register (PMC_PMCCR)

Table 24-2. PMC_PMCCR field descriptions

PRE DIV	CCM	DSM	DDR OFF	CORE OFF	Description
0	0	0	0	0	When Power Architecture core hits breakpoint, enter break mode. When Power Architecture core initiates nap[1] mode or sleep[2] mode entry: undefined, reserved.
0	0	0	0	1	When Power Architecture core hits breakpoint: undefined, reserved. When Power Architecture core initiates nap mode entry: enter nap mode; do not put DRAM in self-refresh mode during nap mode. When Power Architecture core initiates sleep mode entry: enter sleep mode; do not put DRAM in self-refresh mode during sleep mode. PMC_PMCCR is cleared after return to full power mode.
0	0	0	1	1	When Power Architecture core hits breakpoint: undefined, reserved. When Power Architecture core initiates nap mode entry: enter nap mode; put DRAM in self-refresh mode during nap mode. When Power Architecture core initiates sleep mode entry: enter sleep mode; put DRAM in self-refresh mode during sleep mode. PMC_PMCCR is cleared after return to full power mode.
0	0	1	0	0	When Power Architecture core hits breakpoint: undefined, reserved. When Power Architecture core initiates nap mode entry: undefined, reserved. When Power Architecture core initiates sleep mode entry: enter deep sleep mode; PMC_PMCCR is cleared after return to full power mode.
0	1	0	0	0	When Power Architecture core hits breakpoint: undefined, reserved. When Power Architecture core initiates nap mode entry: undefined, reserved. When Power Architecture core initiates sleep mode entry: enter core PLL change mode. PMC_PMCCR is cleared after return to full power mode
1	0	0	0	0	When Power Architecture core hits breakpoint: undefined, reserved. When Power Architecture core initiates nap mode entry: undefined, reserved. When Power Architecture core initiates sleep mode entry: enter Pre divider copy mode. PMC_PMCCR is cleared after return to full power mode.
1	0	0	1	0	When Power Architecture core hits breakpoint: undefined, reserved. When Power Architecture core initiates nap mode entry: undefined, reserved. When Power Architecture core initiates sleep mode entry: put DRAM in self-refresh mode and enter Pre-divider copy mode. PMC_PMCCR is cleared after return to full power mode.
All other values					Undefined, reserved

1 Power Architecture core initiates nap mode entry by setting the e300 NAP bit in the HID register while the POW bit in MSR register is set.

2 Power Architecture core initiates sleep mode entry by setting the e300 SLEEP bit in the HID register while the POW bit in MSR register is set.

24.2.2.2 PMC Event Register (PMC_PMCER)

Address: Base + 0x04 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	WDTO	INT2	INT1
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24-2. PMC Event Register (PMC_PMCER)

Table 24-3. PMC_PMCER field descriptions

Field	Description
INT1	Interrupt Event 1 0 Interrupt event 1 did not occur. 1 Interrupt event 1 condition occurred. This bit is asserted when the Power Architecture core is in nap or sleep mode and another CSB master (i.e. JTAG or tester for MPC5125) initiates CSB transactions without sending interrupt to the core in advance. If enabled an interrupt can be generated in this condition which can be used to wake up the core from nap or sleep mode.
INT2	Interrupt Event 2 0 Interrupt event 2 did not occur. 1 Interrupt event 2 condition occurred. This bit is asserted when the system is about to exit from the core PLL change or PRE_DIV copy mode. Interrupt is generated in this condition which should be used to bring the Power Architecture core back to the full operational mode. The interrupt is always sent in this condition (i.e. non-maskable), but still needs to be enabled in the IPIC so that it reaches the e300 core.
WDTO	DDR Response Watchdog Time Out 0 DDR response watchdog time out did not occur 1 DDR response watchdog time out occurred. While entering deep sleep mode, or while entering nap, sleep or PRE_DIV copy mode if PMC_PMCCR[DDROFF] bit is set, PMC requests the DRAM controller to put DDR into self-refresh mode before placing the e300 core in nap or sleep mode. If no response is received from the DRAM controller in approximately 1 mS, the process is terminated and restarted again until software terminates it. The WDTO bit is set in this scenario for debug purpose. Interrupt is not generated.

24.2.2.3 PMC Interrupt Mask Register (PMC_PMCMR)

Address: Base + 0x08 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INT1E
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24-3. PMC Interrupt Mask Register (PMC_PMCMR)

Table 24-4. PMC_PMCMR field descriptions

Field	Description
INT1E	Interrupt 1 Enable 0 Interrupt is not generated when interrupt event 1 occurs. 1 Interrupt is generated when interrupt event 1 occurs.

24.2.2.4 PMC Shadow Register (PMC_PMCSR)

Address: Base + 0x0C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	PMCSR						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24-4. PMC Shadow Register (PMC_PMCSR)

Table 24-5. PMC_PMCSR field descriptions

Field	Description
PMCSR	Shadow Register. Holds the new CORE_PLL configuration to be used in the core PLL on-the-fly change mode.

24.2.2.5 PMC Wakeup Source Enable Register (PMC_PMCWSE)

The PMC Wakeup Source Enable Register (PMC_PMCWSE) register can be used to enable or disable the external deep sleep mode wakeup signals.

Address: Base + 0x10 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	WSE7	WSE6	WSE5	WSE4	WSE3	WSE2	WSE1	WSE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24-5. PMC Wakeup Source Enable Register (PMC_PMCWSE)

Table 24-6. PMC_PMCWSE field descriptions

Field	Description
WSE7	Deep sleep mode wake up signal enable 7. Reserved. 0 Wake up source is disabled. 1 Wake up source is enabled.
WSE6	Deep sleep mode wake up signal enable 6. RTC. 0 Wake up source is disabled. 1 Wake up source is enabled.
WSE5	Deep sleep mode wake up signal enable 5. MSCAN1_RX. 0 Wake up source is disabled. 1 Wake up source is enabled.
WSE4	Deep sleep mode wake up signal enable 4. MSCAN2_RX. 0 Wake up source is disabled. 1 Wake up source is enabled.
WSE3	Deep sleep mode wake up signal enable 3. GPIO03. 0 Wake up source is disabled. 1 Wake up source is enabled.
WSE2	Deep sleep mode wake up signal enable 2. GPIO02. 0 Wake up source is disabled. 1 Wake up source is enabled.
WSE1	Deep sleep mode wake up signal enable 1. GPIO01 0 Wake up source is disabled. 1 Wake up source is enabled.
WSE0	Deep sleep mode wake up signal enable 0. GPIO00. 0 Wake up source is disabled. 1 Wake up source is enabled.

24.2.2.6 PMC Wakeup Source Polarity (PMC_PMCWSP) Register

The PMC Wakeup Source Polarity (PMC_PMCWSP) register can be used to select polarity of the external deep sleep mode wakeup signals.

Address: Base + 0x14 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	WSP7	WSP6	WSP5	WSP4	WSP3	WSP2	WSP1	WSP0
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Figure 24-6. PMC Wakeup Source Polarity (PMC_PMCWSP) Register

Table 24-7. PMC_PMCWSP field descriptions

Field	Description
WSP7	Deep sleep wake up signal polarity select 7. Reserved. 0 Active low. 1 Active high (default).
WSP6	Deep sleep wake up signal polarity select 6. RTC. 0 Active low. 1 Active high (default).
WSP5	Deep sleep wake up signal polarity select 5. MSCAN1_RX. 0 Active low. 1 Active high (default).
WSP4	Deep sleep wake up signal polarity select 4. MSCAN2_RX. 0 Active low. 1 Active high (default).
WSP3	Deep sleep wake up signal polarity select 3. GPIO03. 0 Active low. 1 Active high (default).
WSP2	Deep sleep wake up signal polarity select 2. GPIO02. 0 Active low. 1 Active high (default).
WSP1	Deep sleep wake up signal polarity select 1. GPIO01. 0 Active low. 1 Active high (default).
WSP0	Deep sleep wake up signal polarity select 0. GPIO00. 0 Active low. 1 Active high (default).

24.3 Functional Description

Based on the four power modes of the embedded low-power e300 Power Architecture CPU core, the module provides five power modes: full-power, doze, nap, sleep, and deep sleep mode. Special sleep modes are provided to allow core PLL reprogramming and pre-divider ratio adjustment. They are called core PLL change mode and pre-divider copy mode. A reference to what is running and what is disabled in the various modes is given in Table 24-8. A reference on how the modes are entered and exited is provided in Table 24-9.

Table 24-8. MPC5125 Power Modes

Mode	e300 state	Periphery clocks	e300 bus snooping	e300 time base registers	PLL, osc state	DRAM state	Wake-up time
Full Power	Running	Running	Active	Active	Active	Active	< 0.1 µS
Doze	Suspended	Running	Active	Active	Active	Active	< 0.1 µS
Nap	Suspended	Running	Suspended	Active	Active	Depends[1,6]	< 0.1 µS[2]
Sleep	Suspended	Running	Suspended	Suspended	Active	Depends[1,6]	< 0.1 µS[2]
Core PLL Change	Suspended	Running	Suspended	Suspended	Active	Active	13,200 IPS clk cycles
PRE_DIV Copy	Suspended	Running	Suspended	Suspended	Active	Depends[3,6]	13,200 IPS clk cycles[4]
Deep Sleep	Suspended	Suspended	Suspended	Suspended	Power-down	Self-refresh[5]	21,200 OSC clk cycles+ 13,200 IPS clk cycle + DRAM wakeup time

1 DRAM state during nap and sleep mode depends on setting of DDROFF bit in the PMC configuration register. When putting DRAM interface in self-refresh mode during nap or sleep, no periphery-generated DRAM access is possible any more, and DRAM transfer initiated by peripherals with master capability (DIU, USB, DMA, and FEC) is unable to take place. If DRAM is left active, the transfer can continue while the core is in sleep mode.

2 To be increased with DRAM wakeup time if DRAM has been configured to self-refresh during nap or sleep mode.

3 DRAM state during pre-divider copy mode depends on setting of DDROFF bit in the PMC configuration register. In case DDR1 or DDR2 memory is used the user can set DDROFF bit so that DRAM goes to self-refresh state.

4 Plus DRAM wakeup time if DDROFF bit is set.

5 DRAM only goes to refresh state when core enters deep sleep mode if DRAM controller has been configured correctly. Please refer to chapter on DRAM controller.

Table 24-9. MPC5125 Power Modes – Sleep and Wake-Up Triggers

Power mode	Events leading to entering this sleep mode	Events leading to return to full power
Doze	Set e300 DOZE bit (HID0[8][1] = 1)while — POW bit in MSR[2] register is set	Interrupt to the e300 Power Architecture core Decrementer interrupt Any reset Machine check exception
Nap with DRAM running	Set e300 NAP bit (HID0[9][1] = 1) while — POW bit in MSR[2] register is set v PMC_PMCCR[3] = 00001	Interrupt to the e300 Power Architecture core Decrementer interrupt Any reset Machine check exception

Table 24-9. MPC5125 Power Modes – Sleep and Wake-Up Triggers (Continued)

Power mode	Events leading to entering this sleep mode	Events leading to return to full power
Nap with DRAM in self-refresh	Set e300 NAP bit (HID0[9][1] = 1) while — POW bit in MSR[2] register is set — PMC_PMCCR[3] = 00011	Interrupt to the e300 Power Architecture core Decrementer interrupt Any reset Machine check exception
Sleep with DRAM running	Set e300 SLEEP bit (HID0[10][1] = 1) while — POW bit in MSR[2] register is set — PMC_PMCCR[3] = 00001	Interrupt to the e300 Power Architecture core Any reset Machine check exception
Sleep with DRAM in self-refresh	Set e300 SLEEP bit (HID0[10][1] = 1) while — POW bit in MSR[2] register is set — PMC_PMCCR[3] = 00011	Interrupt to the e300 Power Architecture core Any reset Machine check exception
Deep sleep	Set e300 SLEEP bit (HID0[10][1] = 1) while — POW bit in MSR[2] register is set — PMC_PMCCR[3] = 00100	Asynchronous interrupt from GPIO, RTC, CAN Power-on reset
Core PLL change mode	Set e300 SLEEP bit (HID0[10][1] = 1) while — POW bit in MSR[2] register is set — PMC_PMCCR[3] = 01000	Interrupt to e300 Power Architecture core Reset Machine check exception
PRE_DIV copy mode (not put DRAM to self-refresh state)	Set e300 SLEEP bit (HID0[10][1] = 1) while — POW bit in MSR[2] register is set — PMC_PMCCR[3] = 10000	Interrupt to e300 Power Architecture core Reset Machine check exception
PRE_DIV copy mode with DRAM in self-refresh	Set e300 SLEEP bit (HID0[10][1] = 1) while — POW bit in MSR[2] register is set — PMC_PMCCR[3] = 10010	Interrupt to e300 Power Architecture core Reset Machine check exception

1 HID0 is a register inside the e300 Power Architecture core.

2 MSR is a register inside the e300 Power Architecture core.

3 PMC_PMCCR register is described in section Section 24.2.2.1, “PMC Configuration Register (PMC_PMCCR).”

NOTE

When the Power Architecture core is in debug mode, the PMC_PMCCR register should be programmed to 0b00000. If the register is programmed to any other value, unpredictable operation can result when the Power Architecture core hits a breakpoint.

24.3.1 Full-Power Mode

This is the default power state of the e300 Power Architecture core. In this power mode, the core is fully powered and the internal functional units are operating at the full-processor clock speed. If the dynamic power management mode is enabled, functional units that are idle automatically enter a low-power state without affecting performance, software execution, or external hardware.

24.3.2 Doze Mode

In doze mode, all functional units of the core are disabled except the time base/decrementer registers and the bus snooping logic.

This mode is entered by programming the doze bit (HID0[8] = 1) when MSR[POW] bit is set. The e300 core enters doze mode several processor clocks after these 2 bits are set.

An asynchronous interrupt, system management interrupt, decrementer interrupt, hard or soft reset, or machine check input (*mcp*) brings the core into the full-power state. The core in doze mode maintains the phase-locked loop (e300 PLL) in a full-power state and locked to the core external clock input (CSB_CLK), so a transition to the full-power state takes only a few processor clock cycles.

The doze mode is an e300 core only low-power mode, the system stays in full-powered mode while the core enters this mode. The core enters and exits doze mode without being controlled by the PMC.

24.3.3 Nap Mode

The nap mode further reduces e300 power consumption by disabling bus snooping, leaving only the time base register and the core PLL in a powered state. While the core enters nap mode, the system stays in full-powered state.

In this mode, bus snooping is disabled. The core waits until the snoop bus is idle before entering nap mode.

When the DDROFF bit is set in the PMC_PMCCR register, described in Section 24.2.2.1, “PMC Configuration Register (PMC_PMCCR),” is set, the DRAM is put into self-refresh mode during nap mode. Setting the DDROFF bit requires the DRAM controller to be configured for putting the DRAM in self-refresh mode upon receipt of the request.

When putting the DRAM in self-refresh mode during nap mode, alternate masters (DIU, FEC, USB, DMA) cannot access DRAM any more. If the DRAM does not enter self-refresh mode, alternate masters can continue reading and writing from DRAM while the core is in a sleep mode.

To enter nap mode, the POW bit in the e300 MSR register must be set, then the COREFOFF bit in the PMC_PMCCR register must be set, before setting the nap bit in an e300 system register (HID0[9] = 1).

The core returns to the full-power state upon receipt of an asynchronous interrupt, system management interrupt, decrementer interrupt, hard or soft reset, or machine check input (*mcp*) signal. A return to full-power state from the nap mode takes only a few processor clock cycles if the DRAM is not put in self-refresh mode. If the DRAM is put in self-refresh mode, the time is dominated by the DRAM wakeup time. When the core is in nap mode, another CSB master (JTAG or tester for MPC5125) may initiate CSB transactions directly without sending an interrupt to the core in advance. A PMC interrupt can be used to wakeup the core in this case if enabled (PMC_PMCER[INT1], see Section 24.2.2.2, “PMC Event Register (PMC_PMCER)”).

24.3.4 Sleep Mode

Sleep mode is an e300 core-only sleep mode. In this mode, the e300 enters sleep mode while the system stays in full-powered mode. The e300 sleep mode reduces the core power consumption to a minimum by disabling all core internal functional units.

In this mode, bus snooping is disabled. The core waits unit the snoop bus is idle before entering sleep mode.

When the DDROFF bit in the PMC_PMCCR register is set, the DRAM is put into self-refresh mode during sleep mode. Setting the DDROFF bit requires the DRAM controller to be configured for putting the DRAM in self-refresh mode upon receipt of the request.

When putting the DRAM in self-refresh mode during sleep mode, alternate masters (DIU, FEC, USB, DMA) cannot access DRAM any more. If the DRAM does not enter self-refresh mode, alternate masters can continue reading and writing from DRAM, while the core is in a sleep mode.

To enter sleep mode, the POW bit in the e300 MSR register must be set, then the COREFOFF bit in the PMC_PMCCR register must be set, before setting the sleep bit in an e300 system register (HID0[10] = 1).

In this mode, the system PLL, core external input clock (CSB_CLK), and e300 PLL are all maintained. The core returns to the full-power state upon receipt of an asynchronous interrupt, system management interrupt, hard or soft reset, or machine check input ($\overline{mcp}$) signal. A return to full-power state from the sleep mode takes only a few processor clock cycles if the DRAM is not put in self-refresh mode. If the DRAM is put in self-refresh mode, the time is dominated by the DRAM wakeup time. When the core is in sleep mode, another CSB master (JTAG or tester for MPC5125) may initiate CSB transfers directly without sending an interrupt to the core in advance. A PMC interrupt can be used to wakeup the core in this case if enabled (PMC_PMCER[INT1], see Section 24.2.2.2, “PMC Event Register (PMC_PMCER)”).

24.3.5 Deep Sleep Mode

The system provides a low-power consumption mode where the e300 core enters sleep mode. The system oscillator, system PLL, and e300 PLL are all powered down and disabled. While the module enters this mode, all internal functional units, except the real-time clock (RTC), are disabled. As the clocks are static, the current draw of the device is reduced to leakage level. The internal state of the device is maintained in deep sleep as long as power is maintained. The real-time clock (RTC) is not disabled in deep sleep mode. If the RTC is used, that portion of the chip continues consuming power in deep sleep mode.

In this mode, bus snooping is disabled. The core waits unit the snoop bus is idle before entering deep sleep mode.

The DRAM is put into self-refresh mode during deep sleep mode. It is required that the DRAM controller is configured for putting the DRAM in self-refresh mode upon receipt of the request.

To enter deep sleep mode, the POW bit in the e300 MSR register must be set, then the DSM bit in the PMC_PMCCR register must be set, before setting the sleep bit in an e300 system register (HID0[10] = 1).

An power-on reset, or an asynchronous interrupt from GPIO, RTC or one of the CAN modules (which occurs when a data transition occurs on the serial input) can be used to bring the module back to the full-power state from deep sleep mode. No clock is required to trigger the wake up process in the case of the GPIO interrupt or the CAN module interrupt. On reception of a wakeup signal from GPIO, RTC, or MSCAN, PMC enables the system oscillator and PLLs in turn and waits for the oscillator to stable and PLLs to lock. After the system clocks are back to full-power mode, the interrupt from GPIO, RTC and MSCAN may bring the e300 core back to the full-power state. The wakeup time is listed in Table 24-8.

During deep sleep mode, state of all peripherals is frozen. All state and data is retained in internal registers, but the peripherals stop reacting on triggers from outside (e.g. a UART does not see incoming data any more). Reinitialization may be needed after leaving deep sleep mode.

- Reinitialization is not needed if the peripheral does not need to resynchronize to an external interface, or if the resynchronization can be done without reinitialization
- Reinitialization is needed if the peripheral requires it to sync again with an external interface.

Reset is not needed after deep sleep mode.

24.3.6 Core PLL Change Mode

This mode can be used to change the core PLL setting. To change it, write the new setting to the shadow register (PMC_PMCSR) in PMC module, and then set the CCM bit in the PMC_PMCCR register. Because the core PLL needs relock, the core must enter sleep mode.

To enter core PLL change mode, the POW bit in the e300 MSR register must be set, then the CCM bit in the PMC_PMCCR register must be set, before setting the sleep bit in an e300 system register (HID0[10] = 1).

In this mode, bus snooping is disabled. The core waits unit the snoop bus is idle before entering core PLL change mode.

After the core enters sleep mode, PMC provides the updated core PLL configuration to the core.

During the core PLL change process, the PMC blocks all interrupts to the core to make sure it is not awakened while the PLL is not locked. The wakeup time is listed in Table 24-8. After the wakeup time, the PMC unblocks the interrupts and the PMC interrupt (PMC_PMCER[INT2], see Section 24.2.2.2, “PMC Event Register (PMC_PMCER)”) can be used to wakeup the core from sleep mode.

24.3.7 PRE_DIV Copy Enable Mode

This mode can change the system clock divide ratio (SYS_DIV), also referred to as pre-divider ratio (PRE_DIV). To change the system clock divide ratio, write the new SYS_DIV value to the shadow register (SCFR2) in the CLOCK module, and then set the PREDIV bit in the PMC_PMCCR register. Because the core PLL needs relock, the core must enter sleep mode. To enter sleep mode, the POW bit in the e300 MSR register must be set, then set the sleep bit (HID0[10] = 1).

In case DDR1 or DDR2 DRAM is used, the DRAM PLL needs to relock because DRAM clock frequency changes. The user can set the PMC_PMCCR[DDROFF] bit so that DRAM goes to self-refresh automatically. It is required that the DRAM controller is configured for putting the DRAM in self-refresh state upon receipt of the request from PMC.

NOTE

Changing pre-divider factor changes all on-chip frequencies. This may upset certain peripherals, and require re-initialization. The device drivers of these peripherals must take care of this.

In this mode, bus snooping is disabled. The core waits unit the snoop bus is idle before entering Core PLL change mode.

After the core enters sleep mode, PMC waits several processor clocks for the core PLL to be switched off, and then asserts a copy_shadow signal to enable the CLOCK module to copy the new SYS_DIV value and update system clock frequency.

During the PRE_DIV copy enable process (i.e. from the point when the e300 core initiates sleep mode entry to when the core PLL is locked to the new system clock input), PMC blocks all interrupts to the core to make sure it's not waked up while the PLL is not locked. The wakeup time is listed in Table 24-8. After the wakeup time, the PMC releases the interrupts and the PMC interrupt (PMC_PMCER[INT2], see Section 24.2.2.2, "PMC Event Register (PMC_PMCER)") can be used to wakeup the core from sleep mode. Then, the system clock divide ratio change is done.

24.3.8 Low-Power Configurations

Table 24-10 summarizes the valid PMC and e300 settings that can be used to put the module into low-power mode, core PLL change mode, or pre-divider copy mode.

Table 24-10. Valid PMC/e300 Low-Power Configurations

Mode to Enter		PMC_PMCCR bit					e300 bit	
PMC	e300	PREDIV	CCM	DSM	DDROFF	COREOFF	MSR[POW]	HID0[8:10]
Doze	Doze	x	x	x	x	x	1	100
Nap	Nap	0	0	0	0	1	1	010
Nap	Nap	0	0	0	1	1	1	010
Sleep	Sleep	0	0	0	0	1	1	001
Sleep	Sleep	0	0	0	1	1	1	001
Deep Sleep	Sleep	0	0	1	0	0	1	001
Core PLL Change	Sleep	0	1	0	0	0	1	001
PRE_DIV Copy	Sleep	1	0	0	0	0	1	001
PRE_DIV Copy	Sleep	1	0	0	1	0	1	001

This page is intentionally left blank.

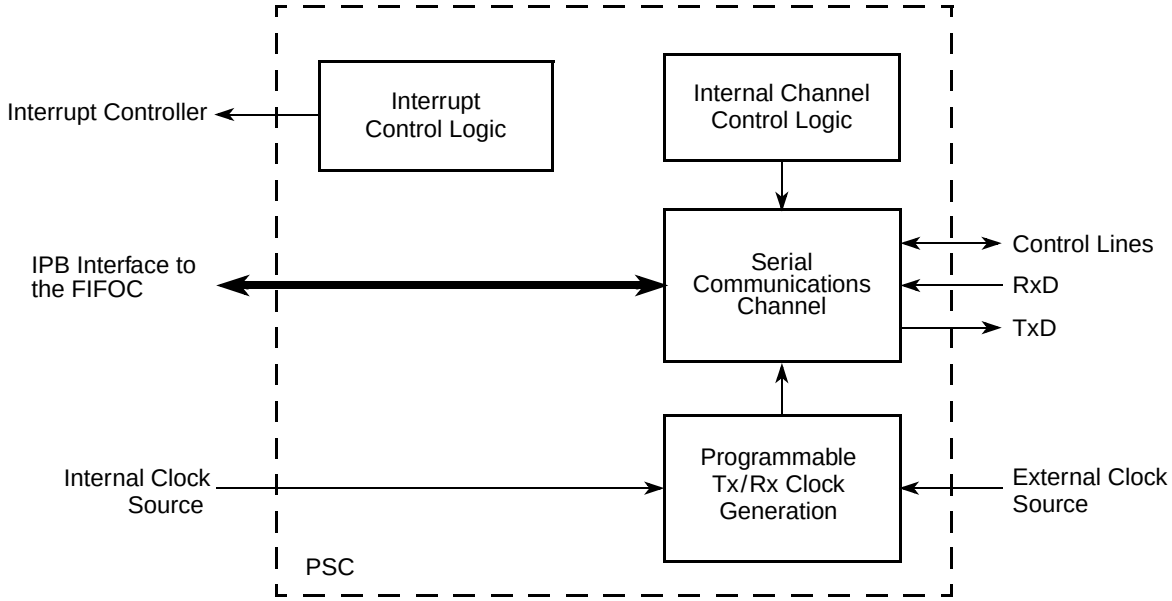
Chapter 25
Programmable Serial Controller (PSC)

25.1 Introduction

Each PSC can be clocked by an internal or external clock source. Figure 25-1 shows a simplified PSC block diagram. In addition, each PSC module interfaces directly to the CPU and consists of the following:

- Serial communication channel
- Programmable transmit (Tx) receive (Rx) clock generation
- Internal channel control logic
- Interrupt control logic

The PSC also provides an MCLK for the external codec, eliminating the need for an external crystal for the external device. For more information about the Codec mode, see Section 25.5.2, “PSC in Codec Mode.”



NOTE: In Codec mode, the number of available TxD and RxD lines depends on the configuration of the Rx-Tx channels fields of register MR2.

Figure 25-1. PSC Block Diagram

25.2 Features

General features:

- Each channel is programmable to normal (full-duplex), automatic echo, local loop-back, or remote loop-back mode
- Automatic wake-up mode for multidrop applications
- Six maskable interrupt conditions

PSC universal asynchronous receiver/transmitter (UART) mode:

- Each is clocked by an internal clock source (IPB clock), eliminating the need for an external crystal
- Full-duplex asynchronous receiver/transmitter channel
- Programmable data format:
  - 5 to 8 data bits plus parity
  - Odd, even, no parity, or force parity
  - 1, 1.5, or 2 stop bits
- Parity, framing, and overrun error detection
- False-start bit detection
- Line-break detection and generation
- Detection of breaks originating in the middle of a character
- Start/end break interrupt/status

PSC codec mode:

- Programmable to interface to an 8-, 12-, 16-, 20-, 24-, or 32-bit codec for soft modem support
- Supports master mode, driving clock, and FrameSync signals
- Supports slave mode, receiving clock, and the FrameSync from the external codec
- Supports multichannel mode, with 3 data lines in codec mode
- Supports full duplex Serial Peripheral Interface (SPI) interface
- Supports I^2S interface
- No parity error, framing error, or line break detection in codec mode
- Ability to generate a master clock (MCLK) for an external codec device, independent from the mode (master or slave)
- Programmable width of the FrameSync signal
- FrameSync and bit clock frequencies are independently programmable
- FrameSync and bit clock polarity are programmable

AC97 mode:

- Supports AC97 mode, only the data slots must be available, the control slot data are generated by the PSC
- Supports AC97 wake-up and power-down modes

25.3 PSC Functions Overview

The PSC module provides different groups of interfaces to connect to different serial devices. Figure 25-2. shows the groups of interfaces.

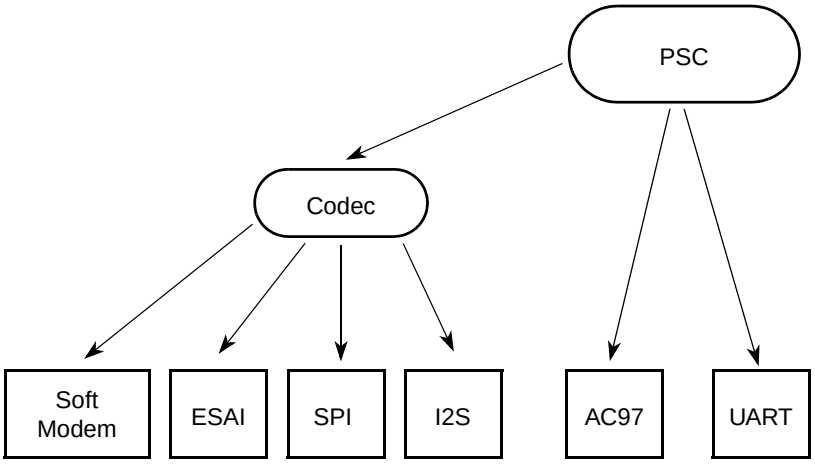


Figure 25-2. PSC Functions Overview

1. **PSC Codec Mode**: In this section, the name codec mode is used as a collective term for the normal soft modem, the SPI, I2S, and ESAI mode. This interface is provided by one internal block of the PSC. The interface consist of serial TX and RX lines, a bit clock line and a FrameSync signal. For these modes, the clock configuration is similar and uses the same configuration registers. The transmitter converts the parallel data from the CPU to a serial bit-stream, and the receiver converts the serial data from the RX line to parallel data. The PSC supports codec mode with 8-, 12-, 16-, 20-, 24-, and 32-bit data width, with active high or active low FrameSync signal, and with programmable bit clock polarity. All codec modes can work as an codec master (PSC dive the bit clock and FrameSync signals) or as a codec slave (PSC receive the bit clock and frame sync signal). For more information about the codec mode, see Section 25.5.2, “PSC in Codec Mode.”
2. **AC97 Mode**: When programmed as AC97, the PSC works as an AC97 controller. This means the PSC receives the BCLK from the external AC97 codec and provides the FrameSync signal to the external codec. If the PSC detects a codec not ready status, the PSC stops sending and receiving data. In AC97 mode, only the used data slots must be in the FIFO. The PSC generates the slot0, slot1, and slot2 values depending on data to send. In AC97 modes, the PSC reads only 32 bits from the FIFO. For more information about the AC97 mode, see Section 25.5.3, “PSC in AC97 Mode.”
3. **PSC UART Mode**: When programmed as a UART, the PSC serial communication channel provides a full-duplex asynchronous receiver and transmitter deriving an operating frequency from an internal clock. The transmitter converts parallel data from the CPU to a serial bit-stream, inserting appropriate start, stop, and parity bits. It outputs the resulting stream on the channel transmitter serial data output (TxD). The receiver converts serial data from the channel receiver serial data input (RxD) to parallel format, checks for start, stop, and parity bits, or line break conditions, and transfers the assembled character onto the bus during read operations. The receiver may be poll-driven or interrupt-driven. For more information about the UART mode, see Section 25.5.1, “PSC in UART Mode.”

Table 25-1. PSC Pin Assignment Versus Operating Mode

	UART	AC97	SPI		Codec TCM/ESAI					I^2S				
			Master	Slave	0 TX, 3 RX	1 TX, 2 RX	1 TX, 1 RX	2 TX, 1 RX	3 TX, 0 RX	0 TX, 3 RX	1 TX, 2 RX	1 TX, 1 RX	2 TX, 1 RX	3 TX, 0 RX
PSCX_0	RTS	BCLK	SCLK		BCLK					SCLK				
PSCX_1	CTS	SYNC	SS		SYNC					LRCK				
PSCX_2	TxD		MOSI	MISO	RxD3	TxD				RxD3	TxD			
PSCX_3	RxD		MISO	MOSI	RxD				TxD3	RxD				TxD3
PSCX_4	DCD	AC97_ RESET	MCLK		RxD2		MCLK	TxD2		RxD2		MCLK	TxD2	

NOTE

For Codec TCM/ESAI and I2S, the BCLK and SCLK pins are outputs when the port is configured as a master and inputs when the port is configured as a slave.

For Codec TCM/ESAI and I2S, the SYNC and LRCK pins are outputs when the port is configured as a master and inputs when the port is configured as a slave.

25.4 Memory Map and Registers

For each PSC, the PSC memory space is composed of a memory block dedicated to PSC registers, a reserved space, and another block dedicated PSC FIFO control (FIFOC) registers. Table 25-2 shows a simplified map of the PSC and FIFOC memory space. Table 25-4 lists the PSC registers.

Table 25-2. PSC and FICOC Memory Map

Address	Register area
0xFF41_1000 (PSC0_BASE)	0x00–0x54: PSC0 registers
	0x58–0x7F: Reserved
	0x80–0xFF: PSC0 FIFOC registers (see Chapter 26, “PSC Centralized FIFO Controller (FIFOC)”)
0xFF41_1100 (PSC1_BASE)	0x00–0x54: PSC1 registers
	0x58–0x7F: Reserved
	0x80–0xFF: PSC1 FIFOC registers
...	**...**

Table 25-2. PSC and FICOC Memory Map (continued)

Address	Register area
0xFF41_1900 (PSC9_BASE)	0x00–0x54: PSC9 registers
	0x58–0x7F: Reserved
	0x80–0xFF: PSC9 FIFOC registers
0xFF41_1A00–0xFF41_1EFF	Reserved
0xFF41_1F00 (FIFOC_BASE)	FIFOC registers (see Chapter 26, “PSC Centralized FIFO Controller (FIFOC)”)

Table 25-3. PSC memory map

Offset from PSC_BASE[1] PSC0: 0xFF41_1000 PSC1: 0xFF41_1100 PSC2: 0xFF41_1200 PSC3: 0xFF41_1300 PSC4: 0xFF41_1400 PSC5: 0xFF41_1500 PSC6: 0xFF41_1600 PSC7: 0xFF41_1700 PSC8: 0xFF41_1800 PSC9: 0xFF41_1900	Register	Access	Reset Value	Section/Page
0x00	Mode Register 1 (MR1)	R/W	—[2]	25.4.1.1/25-665
0x01–0x03	Reserved			
0x04	Mode Register 2 (MR2)	R/W	0x00	25.4.1.2/25-666
0x05–0x07	Reserved			
0x08	Status Register (SR)	R	0x0000	25.4.1.3/25-667
0x0A–0x0B	Reserved			
0x0C	Clock Select Register (CSR)	R/W	0x00	25.4.1.4/25-670
0x10	Command Register (CR)	W	0x00	25.4.1.5/25-671
0x14	Rx Buffer Register (RB)	R	0x0000_0000	25.4.1.6/25-672
0x14	Tx Buffer Register (TB)	W	0x0000_0000	25.4.1.7/25-674
0x18	Input Port Change Register (IPCR)	R	0x00	25.4.1.8/25-675
0x1C	Auxiliary Control Register (ACR)	R/W	0x00	25.4.1.9/25-676
0x20	Interrupt Status Register (ISR)	R	0x0000_0000	25.4.1.10/25-677
0x24	Interrupt Mask Register (IMR)	R/W	0x0000_0000	25.4.1.11/25-678
0x28	Counter Timer Upper Register (CTUR)	R/W	0x00	25.4.1.12/25-680

Table 25-3. PSC memory map (continued)

Offset from PSC_BASE[1] PSC0: 0xFF41_1000 PSC1: 0xFF41_1100 PSC2: 0xFF41_1200 PSC3: 0xFF41_1300 PSC4: 0xFF41_1400 PSC5: 0xFF41_1500 PSC6: 0xFF41_1600 PSC7: 0xFF41_1700 PSC8: 0xFF41_1800 PSC9: 0xFF41_1900	Register	Access	Reset Value	Section/Page
0x2C	Counter Timer Lower Register (CTLR)	R/W	0x00	25.4.1.13/25-680
0x30	Codec Clock Register (CCR)	R/W	—[2]	25.4.1.14/25-681
0x34	AC97 Slots Register (AC97Slots)	R/W	0x0000_0000	25.4.1.15/25-683
0x38	AC97 Command Register (AC97CMD)	R/W	0x0000_0000	25.4.1.16/25-684
0x3C	AC97 Status Data Register (AC97Data)	R	0x0000_0000	25.4.1.17/25-685
0x40	Reserved			
0x44	Input Port Register (IP)	R	—[2]	25.4.1.18/25-686
0x48	Output Port 1 Bit Set (OP1)	R/W	0x00	25.4.1.19/25-686
0x4C	Output Port 0 Bit Set (OP0)	R/W	0x00	25.4.1.20/25-687
0x50	Serial Interface Control Register (SICR)	R/W	0x0000_0000	25.4.1.21/25-687
0x54–0xFF	FIFOC registers. See Chapter 26, “PSC Centralized FIFO Controller (FIFOC).”			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 Mode-dependent. See the register description.

PSC module operation is controlled by writing control bytes into the appropriate registers.

25.4.1 Register Descriptions

NOTE

In the following sections, the terms “assertion” and “negation” are used to avoid confusion between active-low and active-high signals. “Asserted” indicates that a signal is active, independent of the voltage level. “Negated” indicates that a signal is inactive.

25.4.1.1 Mode Register 1(MR1)

The mode registers control configuration.

Address: Base + 0x00 Access: User read/write

	0	1	2	3	4	5	6	7
R	RxRTS	0	Time Out En	PM		PT	B/C	
W								
Reset	0	0	0	0	0	0	0	0

Figure 25-3. Mode Register 1 for UART Mode (MR1)

Address: Base + 0x00 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	1	1	0	0	1	1

Figure 25-5. Mode Register 1 for Other Modes (MR1)

Table 25-4. MR1 field descriptions

Field	Description
RxRTS	**UART** Receiver request-to-send. Allows RTS output to control the CTS input of the transmitting device to prevent receiver overrun. If the receiver and transmitter are incorrectly programmed for RTS control, RTS control is disabled for both. Transmitter RTS control is configured in MR2[TxRTS]. Not used in codec mode. 0 Receiver has no effect on $\overline{\text{RTS}}$. 1 When a valid start bit is received, $\overline{\text{RTS}}$ is negated if the PSC FIFO is full. **Other Modes**. Reserved.
Time Out En	**UART**. Enable the time out counter. If the time out counter is disabled, the TIME_OUT status in the SR was also cleared. 0 Time out counter is disabled 1 Time out counter is enabled **Other Modes**. Reserved.
PM	**UART**. Parity mode. Selects the parity or multi-drop mode for the channel. The parity bit is added to the transmitted character, and the receiver performs a parity check on incoming data. The value of PM affects PT, as shown Table 25-5. PM is not used in codec mode. **Other Modes**. Reserved.
PT	**UART**. Parity Type. PM and PT together select parity type (PM = 0x) or determine whether a data or address character is transmitted (PM = 11). PT is not used in codec mode. See Table 25-5. **Other Modes**. Reserved.

Table 25-4. MR1 field descriptions (continued)

Field	Description
B/C	**UART**. Bits per Character. Selects the number of data bits per character to be sent. The values shown do not include start, parity, or stop bits. B/C is not used in codec mode. 00 5 bits 01 6 bits 10 7 bits 11 8 bits **Other Modes**. Reserved.

Table 25-5. Parity Mode/Parity Type Definitions

PM	Parity Mode	Parity Type (PT=0)	Parity Type (PT=1)
00	With parity	Even parity	Odd parity
01	Force parity	Low parity	High parity
10	No parity	n/a	
11	Multidrop mode	Data character	Address character

25.4.1.2 Mode Register 2 (MR2)

The second register to control the configuration.

Address: Base + 0x04 Access: User read/write

	0	1	2	3	4	5	6	7
R W	CM		TxRTS	TxCTS	SB			
Reset	0	0	0	0	0	0	0	0

Figure 25-6. Mode Register 2 for UART Mode (MR2)

Address: Base + 0x04 Access: User read/write

	0	1	2	3	4	5	6	7
R	CM		0	0	RX -TX Channels			
W								
Reset	0	0	0	0	0	0	0	0

Figure 25-7. Mode Register 2 for Other Modes (MR2)

Table 25-6. MR2 field descriptions

Field	Description
CM	Channel mode. Selects a channel mode. CM is used in UART and codec modes. 00 Normal. 01 Automatic echo. 10 Local loop-back. 11 Remote loop-back.

Table 25-6. MR2 field descriptions (continued)

Field	Description
TxRTS	**UART**. Transmitter ready-to-send. Controls negation of $\overline{RTS}$ to automatically terminate a message transmission. Attempting to program a receiver and transmitter in the same channel for $\overline{RTS}$ control is not permitted and disables $\overline{RTS}$ control for both. TxRTS is not used in codec mode. 0 The transmitter has no effect on $\overline{RTS}$. 1 Setting this bit automatically clears RTS line one bit-time after any characters in the transmitter shift registers are completely sent, including the programmed number of stop bits. **Other Modes**. Reserved.
TxCTS	**UART**. Transmitter clear-to-send. If TxCTS and TxRTS are enabled, TxCTS controls the operation of the transmitter. TxCTS is not used in Codec mode. 0 $\overline{CTS}$ has no effect on the transmitter. 1 Enables clear-to-send operation. The transmitter checks the state of $\overline{CTS}$ each time it is ready to send a character. If $\overline{CTS}$ is asserted, the character is sent. If it is negated, the channel TxD remains in a high state and transmission is delayed until $\overline{CTS}$ is asserted. Changes in $\overline{CTS}$ as a character is being sent do not affect its transmission. **Other Modes**. Reserved.
SB	**UART**. Stop-Bit (length control). Selects the stop bit length Appended to the transmitted character. Stop-bit lengths of 9/16th to 2 bits are programmable for 6- and 8-bit characters. Lengths of 1 1/16th to 2 bits are programmable for 5-bit characters. In all cases, the receiver checks only for a high condition at the center of the first stop-bit position, that is, one bit-time after the last data bit or after the parity bit, if parity is enabled. Therefore, the receiver does not support a stop bit length less than 1. Not used in codec mode, see Table 25-7. **Other Modes**. Reserved.
RX - TX Channels	**Codec**. Define the additional number of used RX and TX channels. Up to 3 RX channels are supported, but the PSC interface only supports three data channels (TX + RX). Therefore, the maximum number of RX channels is also depend on the programmed number of TX channels. 0000 One RX channel, one TX channel. 0001 One RX channel, two TX channels. 0010 Zero RX channel, three TX channels. 0100 Two RX channels, one TX channel. 1000 Three RX channels, zero TX channel. **Other Modes**. Reserved.

Table 25-7. Stop-Bit Lengths

SB	5 Bits	6–8 Bits	SB	5 Bits	6–8 Bits	SB	5–8 Bits	SB	5–8 Bits
0000	1.063	0.563	0100	1.313	0.813	1000	1.563	1100	1.813
0001	1.125	0.625	0101	1.375	0.875	1001	1.625	1101	1.875
0010	1.188	0.688	0110	1.438	0.938	1010	1.688	1110	1.938
0011	1.250	0.750	0111	1.500	1.000	1011	1.750	1111	2.000

25.4.1.3 Status Register (SR)

The read-only SR register shows status of the transmitter, the receiver, and the FIFO.

Address: Base + 0x08 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RB/ EOF	FE	PE	ORE RR	Tx EMP	0	0	0	CDE	Error	Time Out	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-8. Status Register for UART Mode (SR)

Address: Base + 0x08 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	OR ERR	UR ERR	0	0	0	0	Error	0	0	CMD _SEN D	DATA _OVR	DATA _VALI D	UNEX _RX_ SLOT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-9. Status Register for Other Mode (SR)

Table 25-8. SR field descriptions

Field	Description
RB/EOF	**UART**. Received Break. Detects breaks originating in middle of received character. Such a break must persist until the end of next detected character time. 0 No break received. 1 An all-0 character of the programmed length was received without a stop bit.Only a single FIFO position is occupied when a break is received. Further entries to FIFO are inhibited until RxD returns to high state for at least one-half bit-time, which equals two successive PSC clock edges. **Other Modes**. Reserved.
FE/PHYERR	**UART**. Framing Error. Not used (always 0) in codec mode. 0 No framing error occurred. 1 No stop bit detected when corresponding FIFO data character received. Stop bit-check occurs in middle of first stop bit position. **Other Modes**. Reserved.
PE	**UART**. Parity Error. PE is not used (always 0) in codec mode. 0 No parity error occurred. 1 If MR1[PM]equals 0x (with parity or force parity), corresponding FIFO character was received with incorrect parity. If MR1[PM] equals 11 (multidrop), PE stores received A/D bit. **Other Modes**. Reserved.
ORERR	Overrun Error. Indicates whether an overrun occurs. For purposes of overrun, FIFO full means all FIFO space is occupied; the Rx FIFO threshold is irrelevant to overrun. 0 No overrun occurred. 1 One or more characters in Rx data stream were lost. ORERR sets on receipt of a new character when FIFO is full and a character is already in the shift register waiting for an empty FIFO position. When this occurs, the character in the Rx shift register and its break detect, framing error status, and parity error, if any, are lost. ORERR is cleared by the RESET ERROR STATUS command in CR.

Table 25-8. SR field descriptions (continued)

Field	Description
TxEMP/URERR	**UART**. Transmitter Empty. 0 Tx buffer not completely empty. Either a character is being shifted out, or Tx is disabled. Tx is enabled/disabled by programming CR [TC]. 1 Tx has underrun (both the Tx holding register and Tx shift registers are empty). This bit sets after transmission of the last stop bit of a character, if there are no characters in the Tx holding register awaiting transmission. **Other Modes**. Underrun Error. 0 No error. 1 Underrun error occurred, which means the number of Tx FIFO bytes is 0, the Tx shift register is empty, and a FrameSync occurs. In other words, the time has come to transmit a new sample, but no sample is available in the Tx shift register. Unlike UART mode, TxEMP high indicates an error condition similar to the overrun condition (ORERR = 1). It is now cleared the same way as ORERR by a RESET ERROR STATUS command in the CR and not by a reset Tx command in the CR.
CDE/DEOF	**UART**. $\overline{\text{DCD}}$ Status. 0 The $\overline{\text{DCD}}$ input is negated while receiving data. 1 No error. **Other Modes**. Reserved.
Error	Error Status Detect. 0 No error. 1 The PSC controller detected an error state. This error is a combination of the error bits: RB, FE, PE, URERR, ORERR from this register and RX and RX FIFO bit from the TFSTAT and RFSTAT register. **Other Modes**. Reserved.
Time Out	**UART**. Time Out. 0 No Time out event occurred. 1 Time out event occurred. **Other Modes**. Reserved
CMD_SEND	**AC97 Mode**. Command Send Ready. 0 The data in the AC97CMD register was sent out by the AC97 transmitter. 1 The data in the AC97CMD register was not sent out. A write access to the AC97CMD register sets this bit. If the AC97 transmitter sends out the CMD data, this bit is cleared. **Other Modes**. Reserved.
DATA_OVR	**AC97 Mode**. Receive Status Data Overwrite. 0 No received status data overwrite. 1 The received frame contains a new valid data status word in slot 2, but the previous received status data word was not read out before the new one was written to the AC97data register. Therefore, the old status data word was lost. A read access to the AC97 data register clears this bit. **Other Modes**. Reserved.
DATA_VALID	**AC97 Mode**. Received Status Data. 0 The received frame does not contains valid status data. 1 The received frame contains a valid data status word in slot 2. The received data are located in the AC97 data register. A read access to the AC97data register clears this bit. **Other Modes**. Reserved.

Table 25-8. SR field descriptions (continued)

Field	Description
UNEX_RX_SLOT	**AC97 Mode**. Unexpected Receive Slots Detect. 0 The received frame contains the slots defined in the AC97 slots register or a frame without AC97 data (frame is empty or contains only slot 1 or slot 2 data). The case that the receive frame contains all expected data slots plus additional data slots will be also excepted. The RX data of a frame that matches this rule is written to the RX FIFO. 1 The AC97 receiver detects a frame that does not match the rules above. The RX data of the frame that triggers this bit will be ignored. **Other Modes**. Reserved.

25.4.1.4 Clock Select Register (CSR)

The device supports internal and external clocks as source for the UART clock generation. Also the sample rate of the UART controller can be selected in this register.

Address: Base + 0x0C Access: User read/write

	0	1	2	3	4	5	6	7
R	RCS				TCS			
W								
Reset	0	0	0	0	0	0	0	0

Figure 25-10. Clock Select Register for UART Mode (CSR)

Address: Base + 0x0C Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

Figure 25-11. Clock Select Register for Other Modes (CSR)

Table 25-9. CSR field descriptions

Field	Description
RCS	**UART**. Receiver Clock Select Register 0000–1101 Choose the internal IPB Clock with a sample of 16x for the UART receive clock generation. 1110 Choose the external Mclk with a sample of 16x for the UART receive clock generation. 1111 Choose the internal IPB Clock with a sample of 10x for the UART receive clock generation. **Other Modes**. Reserved.
TCS	**UART**. Transmitter Clock Select Register 0000–1101 Choose the internal IPB Clock with a sample of 16x for the UART transmitter clock generation. 1110 Choose the external Mclk with a sample of 16x for the UART transmitter clock generation. 1111 Choose the internal IPB Clock with a sample of 10x for the UART transmitter clock generation. **Other Modes**. Reserved.

25.4.1.5 Command Register (CR)

The command registers (CR) provide the commands to the PSC in all modes. Only multiple commands that do not conflict can be specified in a single write to a CR. For example, reset Tx and enable Tx cannot be specified in one command.

Address: Base + 0x10 Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W		MISC			TC		RC	
Reset	0	0	0	0	0	0	0	0

Figure 25-12. Command Register for all Modes (CR)

Table 25-10. CR field descriptions

Field	Description
MISC	Miscellaneous command. 000 No command. 001 No command. 010 Immediately disables receiver and puts the receiver in a known state. Use this command instead of RECEIVER DISABLE when reconfiguring the receiver. 011 Reset transmitter. In UART mode, this bit immediately disables Tx and clears SR [TxEMP]. No other registers are altered. Because it places the transmitter in a known state, use this command instead of TRANSMITTER DISABLE when reconfiguring the transmitter. In codec mode, the TX prefetch register is cleared and URERR is not cleared by this soft reset. It is cleared the same way as the Rx overflow bit, by a RESET ERROR STATUS command. 100 Reset error status. In UART mode, this command clears ISR[RB,FE,PE,ORERR]. In codec mode, this command clears ISR[ORERR, URERR, DEOF, CMD_SEND, DATA_OVR, DATA_VALID, and UNEX_RX_SLOT]. 101 Reset break change interrupt. Clears the delta break bit, ISR[DB]. Command has no effect in codec mode. 110 Start break. Forces TxD low. — If Tx is empty, break may be delayed as much as one bit-time. — If Tx is active, break starts when character transmission completes. Break is delayed until any character in Tx shift register is sent. Any character in Tx holding register is sent after the break. Tx must be enabled for command to be accepted. This command ignores the $\overline{\text{CTS}}$ state and has no effect in codec mode. 111 Stop break. Causes TxD to go high (mark) within two bit-times. Any characters in the Tx buffer are sent.

Table 25-10. CR field descriptions (continued)

Field	Description
TC	Transmitter command. 00 No action taken. Causes Tx to stay in current mode. — If Tx is enabled, it remains enabled. — If Tx is disabled, it remains disabled. 01 Transmitter enable. Enables operation of Tx channels. SR [TxEMP] sets. If Tx is already enabled, this command has no effect. In UART Mode: TxEMP bits in SR become asserted. In Codec Mode: Tx FIFO can be loaded while Tx is disabled, unlike in UART mode. Therefore this command does not affect URERR behavior. It does not automatically set URERR. If no data is written to Tx FIFO, URERR sets at the first FrameSync after Tx is enabled. In AC97 Mode: URERR sets if Tx FIFO is empty, Tx is enabled, Rx detects a codec ready condition, and a FrameSync occurs before samples are written to the Tx FIFO. **Note:** In codec/AC97 mode, it is not possible to use the transmitter without the receiver. To transmit data only, the receiver must be enabled. 10 Transmitter disable. Terminates Tx operation and clears SR[TxEMP]. — If a character is being sent when Tx is disabled, transmission completes before Tx becomes inactive. — If Tx is already disabled, the command has no effect. In UART Mode: SR[TxEMP] are negated. In Codec Mode: SR[TxEMP] is negated. Tx does not clear unless PSC is in remote loop-back or auto-echo mode. In codec mode, unlike UART mode, the Tx FIFO may be loaded while Tx is disabled. 11 Reserved. Do not use.
RC	Receiver command. 00 No action taken. Causes receiver to stay in current mode. — If receiver is enabled, it remains enabled. — If receiver is disabled, it remains disabled. 01 Receiver enable. Enables receiver. — If PSC module is not in multidrop mode (MR1[PM] ≠ 11), RECEIVER ENABLE command enables channel's receiver and forces it into a search-for-start-bit state. In multidrop mode the Rx continuously monitors the received data regardless of whether it is enabled or not. — If receiver is already enabled, this command has no effect. 10 Receiver disable. Immediately disables receiver. In UART mode any character being received is lost. The command does not affect receiver status bits or other control registers. — If the PSC module is programmed for local loop-back or multidrop mode, the receiver operates even though this command is selected. — If the receiver is already disabled, the command has no effect. In codec mode, if the receiver is disabled while a character is being received, reception completes before the receiver becomes inactive. 11 Reserved. Do not use. **Note:** This field selects a single command.

25.4.1.6 Rx Buffer Register (RB)

Read access from this read only register reads the data directly from the RX shift register. This access bypasses the data in the RX FIFO. Use the data register in the RX FIFO to read the RX data.

Address: Base + 0x14 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RB[0:15]															
W	Used by Tx Buffer															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RB[16:31]															
W	Used by Tx Buffer															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-13. Rx Buffer Register for UARTR/Codec8/16/32 Modes (RB)

Address: Base + 0x14 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RB[0:15]															
W	Used by Tx Buffer															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RB[16:19]				0	0	0	0	0	0	0	0	0	0	0	0
W	Used by Tx Buffer															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-14. Rx Buffer Register for AC97 Mode (RB)

Address: Base + 0x14 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RB[0:15]															
W	Used by Tx Buffer															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RB[16:23]								0	0	0	0	0	0	0	0
W	Used by Tx Buffer															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-15. Rx Buffer Register for Codec24 Mode (RB)

Table 25-11. RB field descriptions

Field	Description
RB	**AC97** (0:19). Received data. AC97 data must be read one complete sample at a time, where all samples except time slot #0 are 20 bits. Time slot #0 data is in bits 0:15. The bits [21:31] are reserved at this mode. **UART/Codec8** (0:31). Received data. For these modes, data can be read 1, 2, or 4 bytes at a time. For 1 byte at a time, all bytes must be read from bits 0:7. For 2 bytes at a time, data must be read from bits 0:15. Lower-bit data was received before upper-bit data. **Codec16** (0:31). Received data. For these modes, data can be read 2 or 4 bytes at a time. For 2 bytes at a time, data must be read from bits 0:15. Lower-bit data was received before upper-bit data. **Codec24** (0:23). Received data. For these modes, data must be read 4 bytes at a time. The lower 24 bits contain the received data word. **Codec32** (0:31). Received data. For these modes, data must be read 4 bytes at a time.
SOF 20	SOF (bit 20) is 1 in the first sample of a new frame, and contains the Start Of Frame indicator. 0 RB[0:19] is not the first sample in the frame. 1 RB[0:19] is the first sample in a new frame. The number 0 slot is called the TAG slot.

25.4.1.7 Tx Buffer Register (TB)

Writing to this register places data directly into the TX shift register. This access bypasses the data in the TX FIFO. Use the data register in the TX FIFO to provide the TX data.

Address: Base + 0x014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Used by Rx Buffer															
W	TB[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Used by Rx Buffer															
W	TB[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-16. Tx Buffer Register for UART/Codec8/16/32 Modes (TB)

Address: Base + 0x014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Used by Rx Buffer															
W	TB[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Used by Rx Buffer															
W	TB[16:19]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-17. Tx Buffer Register for AC97 Mode (TB)

Address: Base + 0x014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Used by Rx Buffer															
W	TB[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Used by Rx Buffer															
W	TB[16:23]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-18. Tx Buffer Register for Codec24 Mode (TB)

Table 25-12. TB field descriptions

Field	Description
TB	**AC97** (0:19). Transmit data. AC97 data must be written one complete sample at a time, where all samples except time slot #0 are 20 bits. Time slot #0 data is in bits 0:15. 0 RB[0:19] is not the first sample in the frame. 1 RB[0:15] is the first sample in a new frame. The number 0 slot is called the TAG slot. The bits [21:31] are reserved at this mode. **AC97** (0:19). Transmit data. AC97 data for the expected time slots (3 to 12). The lower 20 bits contain the valid data word. **UART/Codec8** (0:31). Transmit data. For these modes, data can be written one, two, or four bytes at a time. For one byte at a time, all bytes must be written to bits 0:7. For two bytes at a time, data must be written to bits 0:15. Lower-bit data is stored before upper-bit data. **Codec16** (0:31). Transmit data. For these modes, data can be written two or four bytes at a time. For 2 bytes at a time, data must be written to bits 0:15. Lower-bit data is stored before upper-bit data. **Codec24** (0:23). Transmit data. For these modes, data must be written four bytes at a time. The lower 24 bits contain the valid data word. **Codec32** (0:31). Transmit data. For these modes, data must be written four bytes at a time.

25.4.1.8 Input Port Change Register (IPCR)

The read-only IPCR register shows the current state and change-of-state for the modem control input port.

Address: Base + 0x18 Access: User read-only

	0	1	2	3	4	5	6	7
R	0	0	D_DCD	D_CTS	0	0	DCD	CTS
W								
Reset	0	0	0	0	0	0	0	0

Figure 25-19. Input Port Change Register for UART Modes (IPCR)

Address: Base + 0x18 Access: User read-only

	0	1	2	3	4	5	6	7
R	SYNC	0	D_DCD	D_CTS	0	0	DCD	CTS
W								
Reset	0	0	0	0	0	0	0	0

Figure 25-20. Input Port Change Register for Codec Mode (IPCR)

Table 25-13. IPCR field descriptions

Field	Description
SYNC	**Codec**. Sync detected. 0 Has not detected sync. 1 Detected sync (Frame = 1 in Codec Modes or Sync = 1 in AC97 mode). **Other Modes**. Reserved. A read access to this register clears the SYNC bit
D_DCD	Delta DCD. 0 No change-of-state has occurred since the last time the CPU read the IPCR. A read of the IPCR also clears the IPCR D_DCD bit. 1 A change of state (1/16 or 1 bit duration determined by the CSR, CTUR and CTLR) has occurred at $\overline{\text{DCD}}$ input. When this bit is set, the ACR can be programmed to generate an interrupt to the processor.
D_CTS	Delta CTS. 0 No change-of-state has occurred since the last time the CPU read the IPCR. A read of the IPCR also clears the IPCR D_CTS bit. 1 A change of state, lasting a certain time, has occurred at $\overline{\text{CTS}}$ input. When this bit is set, the ACR can be programmed to generate an interrupt to the processor. After the enable of the PSC, the CPU must read this bit to make sure this bit is cleared at the beginning of the transmission.
DCD	Current state of $\overline{\text{DCD}}$ port. This input is double latched. 0 The current state of the DCD input port is low. 1 The current state of the DCD input port is high.
CTS	Current state of $\overline{\text{CTS}}$ port. This input is double latched. 0 The current state of the $\overline{\text{CTS}}$ input port is low. 1 The current state of the $\overline{\text{CTS}}$ input port is high.

25.4.1.9 Auxiliary Control Register (ACR)

The ACR register controls Tx/Rx handshaking.

Address: Base + 0x1C Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	0
W							IEC1	IEC0
Reset	0	0	0	0	0	0	0	0

Figure 25-21. Auxiliary Control Register for all Modes (ACR)

Table 25-14. ACR field descriptions

Field	Description
IEC1	Interrupt Enable Control for D_DCD. 0 D_DCD has no effect on the IPC in the ISR. 1 When the D_DCD becomes high, IPC bit in the ISR sets (causing an interrupt if mask is set).
IEC0	Interrupt Enable Control for D_CTS. 0 D_CTS has no effect on the IPC in the ISR. 1 When the D_CTS becomes high, IPC bit in the ISR sets (causing an interrupt if mask is set). After enabling the PSC, the D_CTS bit can be set. Therefore, it's important to clear the D_CTS bit before enabling this interrupt.

25.4.1.10 Interrupt Status Register (ISR)

The read-only ISR register provides status for all potential interrupt sources. Register contents are masked by the IMR.

- If an ISR flag sets and the corresponding IMR bit is also set, the internal interrupt output is asserted.
- If the corresponding IMR bit is cleared, the ISR bit state has no effect on the interrupt output.

Address: Base + 0x2C Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPC	0	0	ORERR	TxEMP	DB	0	0	0	Error	Time Out	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-22. Interrupt Status Register for UART Mode (ISR)

Address: Base + 0x2C Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPC	0	0	ORERR	URERR	0	0	0	0	Error	0	0	CMD_SEND	DATA_OVR	DATA_VALID	UNEX_RX_SLOT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-23. Interrupt Status Register for Other Modes (ISR)

Table 25-15. ISR field descriptions

Field	Description
IPC	Input port change interrupt. 0 No IPC event has occurred. 1 An IPC event has occurred.
ORERR	Overrun Error. This bit is identical to the ORERR bit in the SR register. To clear this interrupt use the reset error status command in the CR register.

Table 25-15. ISR field descriptions (continued)

Field	Description
TxEMP/ URERR	**UART**. TxEMP. This bit is identical to the URERR bit in the SR register. **Other Modes**. Underrun Error. This bit is identical to the URERR bit in the SR register. To clear this interrupt use the reset error status command in the CR register.
DB	**UART**. Delta Break. Receiver detect an Delta Break state. **Other Modes**. Reserved
Reserved	**Other Modes**. Reserved.
Error	Error. This bit is identical to the error bit in the SR register. To clear this interrupt, use the reset error status command in the CR register.
Time Out	**UART**. Time Out This bit is identical to the TimeOut bit in the SR register. **Other Modes**. Reserved
CMD_SEND	**AC97 Mode**. Command Send ready This bit is identical to the CMD_SEND bit in the SR register. To clear this interrupt use the reset error status command in the CR register. **other Modes**. Reserved.
DATA_OVR	**AC97 Mode**. Receive Data Overwrite This bit is identical to the DATA_OVR bit in the SR register. To clear this interrupt, use the reset error status command in the CR register. **Other Modes**. Reserved.
DATA_VALID	**AC97 Mode**. Received Status Data This bit is identical to the DATA_VALID bit in the SR register. To clear this interrupt, use the reset error status command in the CR register. **Other Modes**. Reserved.
UNEX_RX_SLOT	**AC97 Mode**. Unexpected RX Slots detect This bit is identical to the UNEX_RX_SLOT bit in the SR register. To clear this interrupt, use the reset error status command in the CR register. **Other Modes**. Reserved.

25.4.1.11 Interrupt Mask Register (IMR)

The IMR register selects corresponding bits in the ISR that cause an interrupt.

- If one ISR bit is set and the corresponding IMR bit is also set, the internal interrupt output is asserted.
- If the corresponding bit in IMR is 0, the state of the ISR bit has no effect on the interrupt output. The IMR does not mask reading the ISR.

Address: Base + 0x24 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPC	0	0	ORERR	TxEMP	DB	0	0	0	Error	Time Out	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-24. Interrupt Mask Register for UART Mode (IMR)

Address: Base + 0x24 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPC	0	0	ORERR	URERR	0	0	0	0	Error	0	0	CMD_SEND	DATA_OVR	DATA_VALID	UNEX_RX_SLOT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-25. Interrupt Mask Register for Other Modes (IMR)

Table 25-16. IMR field descriptions

Field	Description
IPC	Input port change interrupt. 0 IPC has no effect on the interrupt. 1 Enable the interrupt for IPC in the ISR register.
ORERR	Overrun Error. 0 ORERR has no effect on the interrupt. 1 Enable the interrupt for ORERR.
TxEMP/ URERR	**UART**. TxEMP. 0 TxEMP has no effect on the interrupt. 1 Enable the interrupt for TxEMP. **Other Modes**. Underrun Error. 0 URERR has no effect on the interrupt. 1 Enable the interrupt for URERR.
DB	**UART**. Delta Break. 0 DB has no effect on the interrupt. 1 Enable the interrupt for DB. **Other Modes**. Reserved.
Reserved	**Other Modes**. Reserved.
Error	Error. 0 Error bit in the ISR register has no effect on the interrupt. 1 Enable the interrupt for Error.
Time Out	**UART**. Time Out. 0 Time Out has no effect on the interrupt. 1 Enable the interrupt for Time Out. **Other Modes**. Reserved.
CMD_SEND	**AC97 Mode**. Command Send ready. 0 CMD_SEND bit in the ISR register has no effect on the interrupt. 1 Enable the interrupt for CMD_SEND. **Other Modes**. Reserved.

Table 25-16. IMR field descriptions (continued)

Field	Description
DATA_OVR	**AC97 Mode**. Receive Data Overwrite. 0 DATA_OVR bit in the ISR register has no effect on the interrupt. 1 Enable the interrupt for DATA_OVR. **Other Modes**. Reserved.
DATA_VALID	**AC97 Mode**. Received Status Data. 0 DATA_VALID bit in the ISR register has no effect on the interrupt. 1 Enable the interrupt for DATA_VALID. **Other Modes**. Reserved.
UNEX_RX_SLOT	**AC97 Mode**. Unexpected RX Slots detect. 0 UNEX_RX_SLOT bit in the ISR register has no effect on the interrupt. 1 Enable the interrupt for UNEX_RX_SLOT. **Other Modes**. Reserved.

25.4.1.12 Counter Timer Upper Register (CTUR)

This register holds the upper bytes of the preload value used by the timer to provide a given baud rate. Reading from this register shows the current value of the baud rate generation counter. For a detailed description, see Section 25.4.1.13, “Counter Timer Lower Register (CTLR).”

Address: Base + 0x28 Access: User read/write

	0	1	2	3	4	5	6	7
R W	CTUR[0:7]							
Reset	0	0	0	0	0	0	0	0

Figure 25-26. Counter Timer Upper Register for all Modes (CTUR)

Table 25-17. CTUR field descriptions

Field	Description
CTUR	**Code**. Frame Sync width, define the number of bit clocks during the FrameSync signal is active. FrameSync Width equals CTUR[0:7] + 1. **UART/ SPI**. Baud rate prescaler value. See Section 25.4.1.13, “Counter Timer Lower Register (CTLR).” **Other**. Reserved.

25.4.1.13 Counter Timer Lower Register (CTLR)

This register holds the lower bytes of the preload value used by the timer to provide a given baud rate. Reading from this register shows the current value of the baud rate generation counter.

Address: Base + 0x2C Access: User read/write

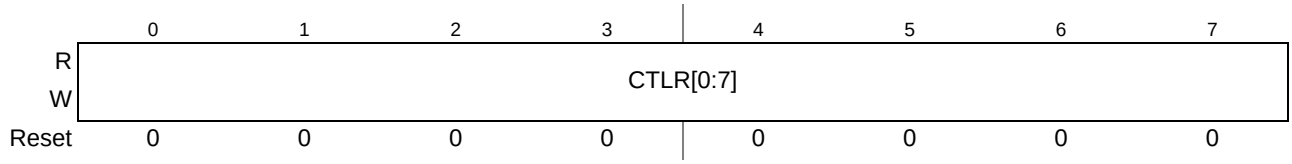


Figure 25-27. Counter Timer Lower Register for all Modes (CTLR)

Table 25-18. CTLR field descriptions

Field	Description
CTLR	**UART**. Baud rate prescale value. The baud rate is calculated as: $\text{Baud rate} = \frac{\text{IPB clock frequency}}{\text{CT[0:15] x prescaler}}$ where: CT[0:15] = {CTUR[0:7], CTLR[0:7]} The minimum CT value is 1; 0 denotes counter stop. The prescaler was defined in the CSR register. **SPI**. Delay After Transfer (DTL). When the PSC is in SPI mode (SICR[SPI] = 1), the counter timer determine the length of time the PSC delays after each serial transfer (the length of time that SS stays high/inactive between consecutive transfers). This feature exists in a QSPI. Delay after transfer can be used to ensure the deselect time requirement (for peripherals having such a requirement) is met. Some peripherals must be deselected for a minimum period of time between consecutive serial transfers. A delay after transfer can be inserted between consecutive transfers to a given peripheral to ensure its minimum deselect time requirement is met or to allow serial A/D converters to complete conversion before the next transfer is made. $\text{DTL} = \frac{\text{CT[0:15]} + 2}{\text{IPB clock frequency}} + \frac{3}{\text{MCLK frequency}}$ where: CT[0:15] = {CTUR[0:7], CTLR[0:7]} **Other**. Reserved.

25.4.1.14 Codec Clock Register (CCR)

This register defines the divider for the FrameSync and BCLK generation for codec mode. This register value has effect only when the GenClk bit in the PSC control register SICR is set or the UART mode was selected.

Address: Base + 0x030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FrameSyncDiv[0:7]								BCLKDiv[0:7]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BCLKDiv[8:15]								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-28. Codec Clock Register for Codec Mode (CCR)

Address: Base + 0x030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TimeOut[0:7]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TimeOut[8:15]								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-29. Codec Clock Register for UART Modes (CCR)

Table 25-19. CCR field descriptions

Field	Description
FrameSyncDiv	**Codec**. Frame Sync Divider. FrameSync is generated internally by dividing down the bit clock. The FrameSyncDiv defines the number of bit clock cycles between two active frame edges: FrameSync Length = FrameSyncDiv[0:7] + 1 For more information, see Section 25.5.2.3, “Transmitting and Receiving in Soft Modem Codec Mode.” **Codec / SPI**. Delay before SCK (DSCKL). When the PSC is in SPI mode (SICR[SPI] = 1), the FrameSyncDiv divider is used to determine the length of time the PSC delays after SS goes low/active before the first SCK transition of the serial transfer. This is a feature that exists in a QSPI. The following equation determines the actual delay before SCK: $\text{DSCKL delay} = \frac{\text{FrameSyncDiv[0:7]} + 1)}{\text{MCLK Frequency}}$ **Other Modes**. Reserved. The value 0x00 stops this counter and disables the clock generator.

Table 25-19. CCR field descriptions (continued)

Field	Description
BCLKDiv	**Codec**. Bit Clock Divider. Bit clock is generated internally by dividing down the MCLK frequency as follows: $\text{BCLK frequency} = \frac{\text{MCLK Frequency}}{\text{BCLKDiv[0:15]} + 1}$ **Codec SPI**. Baud rate. SCK is generated internally by dividing down the MCLK frequency as follows: $\text{SCK frequency} = \frac{\text{MCLK Frequency}}{\text{BCLKDiv[0:15]} + 1}$ The minimum BCLKDiv for SPI mode is 3.
TimeOut	**UART Modes**. Time Out count value. TimeOut[0:15] define the number of UART clock events before the Time_Out event occurred if enabled. **Other Modes**. Reserved.

25.4.1.15 AC97 Slots Register (AC97Slots)

This register defines which slots are expected in a receive AC97 frame and which slots are sent in a AC97 TX frame. If the received frame does not match the expected slots, the SR[UNEXP_RX_SLOTS] bit is set. This register has an effect only if the AC97 mode is selected in the SICR register and if the EnAC97 bit is set to 1.

Address: Base + 0x34 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	TX_SLOTS[3:12]									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	RX_SLOTS[3:12]									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-30. AC97 Slots Register (AC97Slots)

Table 25-20. AC97Slots field descriptions

Field	Description
TX_Slots	**AC97 Mode**. Expected Transmit Slots The bits in this register specify which data slots [3:12] are sent in an AC97 TX frame. Each bit represents one data slot. The AC97 transmitter uses this information to generate the slot0 and reads out the according number of data words from the TXFIFO. If the TXFIFO is empty, an empty AC97 frame is sent until new data is available. **Other Modes**. Reserved.
RX_Slots	**AC97 Mode**. Expected Receive Slots The bits in this register specifies which data slots [3:12] in the receive AC97 frame must contain valid data. The AC97 codec selects the valid data slots by setting the according data valid bit in slot0[12:3]. Each bit represents one data slot. If the received valid slots do not match the expected slots, the unexpected slot received state occurs. See register SR. The received data is written to the RXFIFO only if the received slots matched the expected slots. **Other Modes**. Reserved.

25.4.1.16 AC97 Command Register (AC97CMD)

This register contains the AC97 address for transmit slot1 and the AC97 command data for transmit slot 2. A write access to any byte of this register sets the SR[CMD_SEND] bit to one. The AC97 transmitter generates a frame with valid slot1 and slot2 and pastes the values of this register to the next transmitted slot1 and slot2. If the data was sent, the SR[CMD_SEND] bit is cleared by the transmitter.

Address: Base + 0x38 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	A97 CMD	AC97 Control Register Index							AC97 Command Data[15:8]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AC97 Command Data[7:0]								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-31. AC97 Command Register (AC97CMD)

Table 25-21. AC97CMD field descriptions

Field	Description
AC97 CMD	**AC97 Mode**. AC97 Command This bit indicates if the access to the control register is a read or write access. It is pasted to the slot1 bit 19. 0 Write access 1 Read access **other Modes**. Reserved.
AC97 Control Register Index	**AC97 Mode**. AC97 Address Register This register contains target control register address. It is pasted to the slot1 bit 18 to 12. **Other Modes**. Reserved.
AC97 Command Data	**AC97 Mode**. AC97 Command Data Register This register defines the command data value for a write command. It is pasted to the slot2 bit 19 to 4. **Other Modes**. Reserved.

25.4.1.17 AC97 Status Data Register (AC97Data)

This read-only register contains the received response of an AC97 read command. If this register contains new data, the SR[DATA_VALID] is set to 1 by the receiver. A read access to this register clears the SR[DATA_VALID] bit.

Address: Base + 0x3C Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	AC97 Register Index Echo							AC97 Control Register Read Data[15:8]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AC97 Control Register Read Data[7:0]								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-32. AC97 Status Data Register (AC97Data)

Table 25-22. AC97Data field descriptions

Field	Description
AC97 Register Index Echo	**AC97 Mode**. AC97 Register Index Echo This register contains the received register index echo from the RX slot 0. **Other Modes**. Reserved.
AC97 Control Register ReadData	**AC97 Mode**. AC97 Control Register Read Data This register contains the received control data from Rx slot 2. **Other Modes**. Reserved.

25.4.1.18 Input Port Register (IP)

This read-only IP register shows the current state of the input ports.

Address: Base + 0x44 Access: User read-only

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	DCD	CTS
W								
Reset	1	1	1	1	1	1	0	0

Figure 25-33. Input Port Register for UART Modes (IP)

Address: Base + 0x44 Access: User read-only

	0	1	2	3	4	5	6	7
R	0	TGL	0	0	0	0	DCD	CTS
W								
Reset	1	0	1	1	1	1	0	0

Figure 25-34. Input Port Register for Codec Mode (IP)

Address: Base + 0x44 Access: User read-only

	0	1	2	3	4	5	6	7
R	LPWR	TGL	0	0	0	0	DCD	CTS
W								
Reset	1	1	1	1	1	1	0	0

Figure 25-35. Input Port Register for AC97 Mode (IP)

Table 25-23. IP field descriptions

Field	Description
LPWR	**AC97**. Low-power mode in AC97 mode 0 Codec is in low power mode. 1 Usual operation. **Other Modes**. Reserved
TGL	**AC97 / Codec**. Test usage. Toggle by FrameSync. **Other Modes**. Reserved.
DCD	Current state of the $\overline{\text{DCD}}$ input. 0 $\overline{\text{DCD}}$ input is low. 1 $\overline{\text{DCD}}$ input is high.
CTS	Current state of the $\overline{\text{CTS}}$ input 0 Input port $\overline{\text{CTS}}$ is low. 1 Input port $\overline{\text{CTS}}$ is high.

25.4.1.19 Output Port 1 Bit Set (OP1)

Output ports are asserted by writing to this register, read back the current value of the ports.

Address: Base + 0x48 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	RES	RTS
W								
Reset	0	0	0	0	0	0	0	0

Figure 25-36. Output Port 1 Bit Set Register for all Modes (OP1)

Table 25-24. OP1 field descriptions

Field	Description
RES	Assert RES output. 0 No operation 1 Asserts output port RES, (RES becomes 0).
RTS	**AC97**. Reserved **Other Modes**. Assert RTS output. 0 No operation 1 Asserts output port $\overline{\text{RTS}}$, ($\overline{\text{RTS}}$ becomes 0).

25.4.1.20 Output Port 0 Bit Set (OP0)

Output ports are negated by writing to this register, read back the current value of the ports.

Address: Base + 0x4C Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	RES	RTS
W								
Reset	0	0	0	0	0	0	0	0

Figure 25-37. Output Port 0 Bit Set Register for all Modes (OP0)

Table 25-25. OP0 field descriptions

Field	Description
RES	Assert RES output. 0 No operation 1 Negates output port RES, (RES becomes 1).
RTS	**AC97**. Reserved **Other Modes**. Assert $\overline{\text{RTS}}$ output. 0 No operation 1 Negates output port $\overline{\text{RTS}}$, ($\overline{\text{RTS}}$ becomes 1).

25.4.1.21 Serial Interface Control Register (SICR)

This register sets the main operation mode.

Address: Base + 0x50 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ACRB	AWR	DTS1	SHDIR	SIM[3:0]				GenClk	I2S	ClkPol	SyncPol	0	0	ESAI	EnAC97
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SPI	MSTR	CPOL	CPHA	UseEOF	0	0	En_OutBuf	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-38. Serial Interface Control Register for all Modes (SICR)

Table 25-26. SICR field descriptions

Field	Description
ACRB	**AC97**. AC97 Cold Reset to the transceiver in PSC. This bit was prepared for backward compatibility with the MCF5407 USART. It is recommended to use OP1 and OP0 registers to set and to reset AC97 reset line. 0 The transceiver recovers from low power mode in AC97. 1 The transceiver stays in the current state. **Other Modes**. Reserved.
AWR	**AC97**. AC97 Warm Reset (to the PSC and off-chip AC97 Codec) 0 AC97 warm reset is negated. $\overline{\text{RTS}}$ output functions normally as the AC97 FrameSync. 1 Force 1 on $\overline{\text{RTS}}$ output, which is used as the AC97 FrameSync, and the PSC recovers from AC97 power down mode. **Other Modes**. Reserved.
DTS1	**Codec**. Delay of time slot #1. 0 First bit of first time slot of a new frame starts at the rising edge of FrameSync. 1 First bit of first time slot of a new frame starts one bit clock cycle after the rising edge of FrameSync. **Other Modes**. Reserved.
SHDIR	**Codec**. Shift Direction. 0 MSB first. 1 LSB first. **Other Modes**. Reserved.
SIM[3:0]	PSC operation mode. **CAUTION**: When the operating mode change occurs, all Rx/Tx and error statuses are reset. Rx and Tx are disabled. 0000 = UART mode, $\overline{\text{DCD}}$ input ignored. 1000 = UART mode, $\overline{\text{DCD}}$ input is effective. 0001 = Codec mode, 8-bit data. 1001 = Codec mode, 12-bit data. 0010 = Codec mode, 16-bit data. 1010 = Codec mode, 20-bit data. x011 = AC97 mode. 0100 = Reserved. 1100 = Reserved. x101 = Reserved. x110 = Reserved. 0111 = Codec mode, 24-bit data. 1111 = Codec mode, 32-bit data.

Table 25-26. SICR field descriptions (continued)

Field	Description
GenClk	**Codec**. Generate Bit Clock and FrameSync. Not used to enable the SPI master mode, but this bit must also be set to the MSTR to enable the SPI master mode. 0 Use bit clock and FrameSync provided by external device. 1 Use bit clock and FrameSync generated internally from MCLK. **Other Modes**. Reserved.
I2S	**Codec**. I2S mode 0 No I2S mode supported. 1 PSC works in I2S mode. **Other Modes**. Reserved.
ClkPol	**Codec**. Bit Clock Polarity 0 Data in is sampled on the falling edge of the BCLK and data out is shifted on the rising edge. 1 Data in is sampled on the rising edge of the BCLK and data out is shifted on the falling edge. **Other Modes**. Reserved. **Note:** This bit must be cleared during SPI mode.
SyncPol	**Codec**. FrameSync Polarity, must be cleared during SPI mode 0 FrameSync is low true. 1 FrameSync is high true. **Codec I2S**. FrameSync Polarity. 0 Frame starts if LRCK is low. 1 Frame starts if LRCK is high. **Other Modes**. Reserved.
ESAI	**Codec**. Enhanced Serial Audio Interface 0 PSC does not support the ESAI mode. 1 PSC supports the ESAI mode. This mode allows the PSC to send and receive more the one data word per frame, if the frame length is greater than the word length. The PSC send only complete data words. **Other Modes**. Reserved.
EnAC97	**Codec**. Normal AC97 mode. Takes effect only when the AC97 mode is selected. (SIM = 0x3) 0 If AC97 mode was selected, Legacy AC97 mode used. 1 If AC97 mode was selected, AC97 mode transmits and receives the data. **Other Modes**. Reserved.
SPI	**Codec**. SPI mode. 0 PSC does not behave like an SPI. 1 PSC behaves like an SPI. **Other Modes**. Reserved.
MSTR	**Codec**. SPI Master mode. Takes effect only when bit SICR[SPI mode] equals 1. Also, the GenClk bit must be set to enable the clock generation behavior of the SPI master mode. 0 PSC behaves as an SPI slave. 1 PSC behaves as an SPI master. **Other Modes**. Reserved.
CPOL	**Codec**. SPI Clock Polarity. Takes effect only when bit SICR[SPI mode] equal 1. This bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. 0 Active-low clocks selected; SCK idles high. 1 Active-high clocks selected; SCK idles low. **Other Modes**. Reserved.

Table 25-26. SICR field descriptions (continued)

Field	Description
CPHA	**Codec**. SPI Clock Phase This bit is used to shift the SCK serial clock. To transmit data between SPI modules, the SPI modules must have identical CPHA values. 0 Data transfer starts which assertion of $\overline{SS}$. 1 Data transfer starts with the first edge of SCK. **Other modes**. Reserved.
UseEOF	**Codec**. Use End-of-Frame flag takes effect only when bit 16 SPI mode equals 1. 0 One, two, or four bytes are transferred while slave select (SS) is held low, as determined by Codec8, Codec16, Codec24, or Codec32 selected by SICR[SIM]. 1 Multiple bytes are transferred while maintaining SS low, up to and including the next byte read from the Tx FIFO that has its EOF flag set. **Other modes**. Reserved.
En_OutBuf	Enable Output Buffer. 0 The output buffer for the 5 PSC pads are enabled if the TX or the RX part was enabled. 1 The output buffer for the 5 PSC pads is enabled independent from the transmitter or receiver.

25.5 Modes of Operation

This section describes the different PSC operation modes, including the pin muxing, the module configuration, signal definition, and some programming examples. All PSCs are independent and can be used at the same time in different modes.

25.5.1 PSC in UART Mode

Select the UART mode by writing the corresponding value to the PSC Control (SICR) register. The PSC UART mode is the default mode after reset. The important registers to configure the PSC for UART mode are:

- SICR register — select the UART mode
- CSR register — select the clock source
- CTUR, CTLR register — select the baud rate
- MR1 register — select the UART mode (parity mode, bits per character)
- MR2 register — select $\overline{RTS}$ and $\overline{CTS}$ control, Stop Bit Length
- CR register — enable or disable receiver and transmitter

25.5.1.1 Block Diagram and Signal Definition for UART Mode

Figure 25-39 shows a simplified block diagram of the PSC for UART mode.

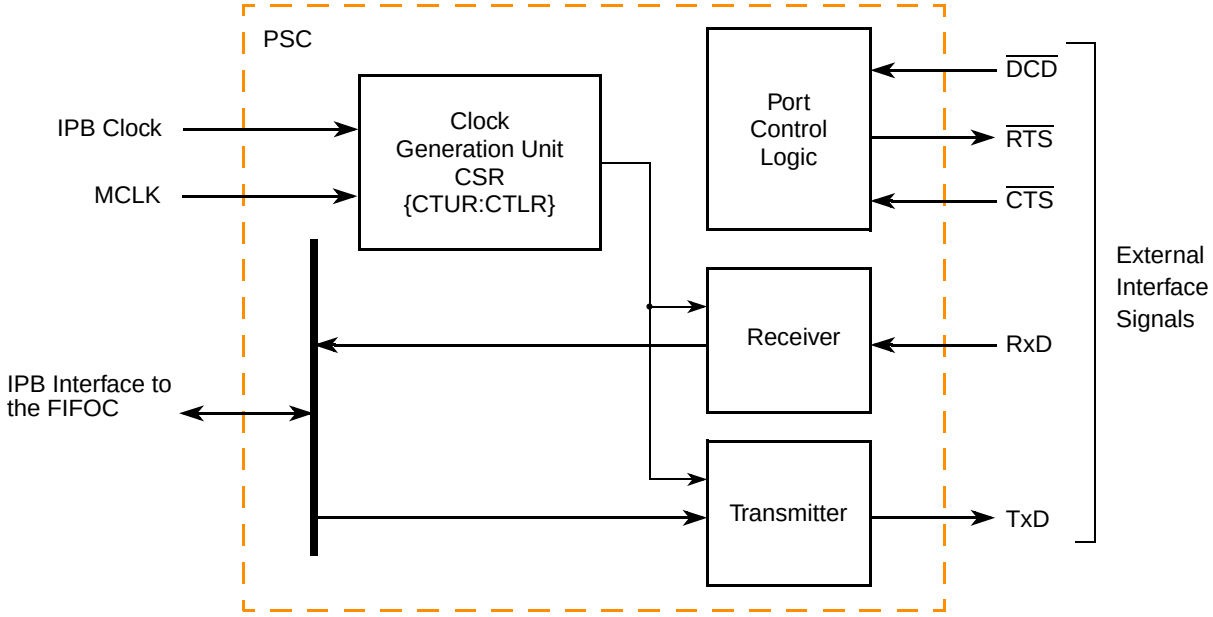


Figure 25-39. PSC UART Block Diagram

An internal interrupt request signal is provided to notify the interrupt controller of an interrupt condition. The output is the logical NOR of unmasked ISR bits. The interrupt level of a PSC module is programmed in the interrupt controller.

The PSC can automatically transfer data using the DMA, rather than interrupting the core.

Table 25-27 briefly describes the PSC module signals.

NOTE

The terms assertion and negation are used to avoid confusion between active-low and active-high signals.

- Asserted indicates a signal is active, independent of the voltage level
- Negated indicates a signal is inactive.

Table 25-27. PSC Signal Description for UART Mode

Signal	Description
TxD	Transmitter Serial Data Output. TxD is held high (mark condition) when Tx is disabled, idle, or operating in the local loop-back mode. Data is shifted out on TxD on the falling edge of the clock source, with the least significant bit (LSB) sent first.
RxD	Receiver Serial Data Input. Data received on RxD is sampled on the rising edge of the clock source, with the LSB received first.
CTS	Clear-to-Send. This input can generate an interrupt on a change of state.
RTS	Request-to-Send. This output can be programmed to be negated or asserted automatically by Rx or Tx. When connected to a transmitter $\overline{CTS}$, $\overline{RTS}$ can control serial data flow.
DCD	Data carrier detect Input. In the enhanced UART mode, this signal must be asserted during the data transmission.

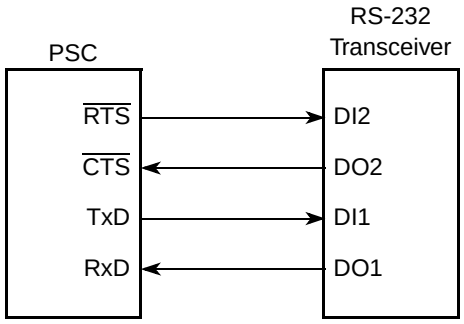


Figure 25-40. Signal Configuration for a PSC/RS-232 Interface

25.5.1.2 UART Clock Generation

The supported UART Baud rate depends on the input frequency of the selected clock source, the selected divide ratio and the selected sample rate. The clock source and the sample rate are defined in the CSR register. The divide ratio for the divider of the internal clock source is defined by concatenated CTUR and CTLR registers. For more information see Figure 25-41.

Calculation of the UART Baud rate based on the internal clock:

Eqn. 25-1

$$\text{Baud rate} = \frac{\text{IPB Clock}}{\text{sample rate x divider \{CTUR:CTLR\}}}$$

Calculation of the UART Baud rate based on external MCLK:

Eqn. 25-2

$$\text{Baud rate} = \frac{\text{MCLK}}{\text{sample rate}}$$

For example, to generate a 9600 baud rate based on an internal 66 MHz IP bus frequency and a sample rate of 16, the register values can be calculated as follows:

$$\text{Divider} = \frac{66\ \text{MHz}}{16 \times 9600} = 430(\text{decimal}) = 0\text{x}00\text{D}7$$

Therefore, CTUR equals 0x00 and CTLR equals 0xD7.

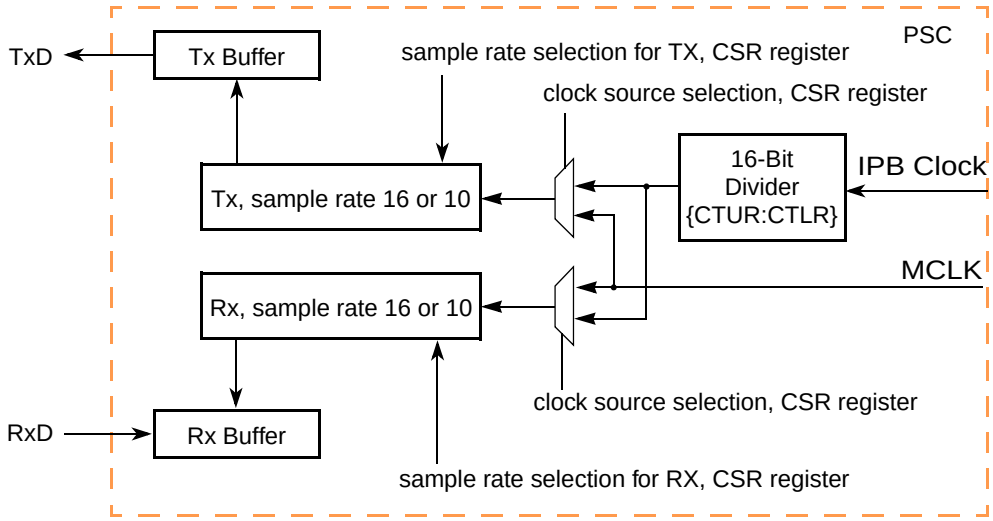


Figure 25-41. Clocking Source Diagram

25.5.1.3 Transmitting in UART Mode

After a hardware reset, all PSCs are in UART mode. The PSC command register (CR) enables the transmitter. The transmitter converts parallel data from the CPU to a serial bit-stream on TxD. It automatically sends a start bit followed by:

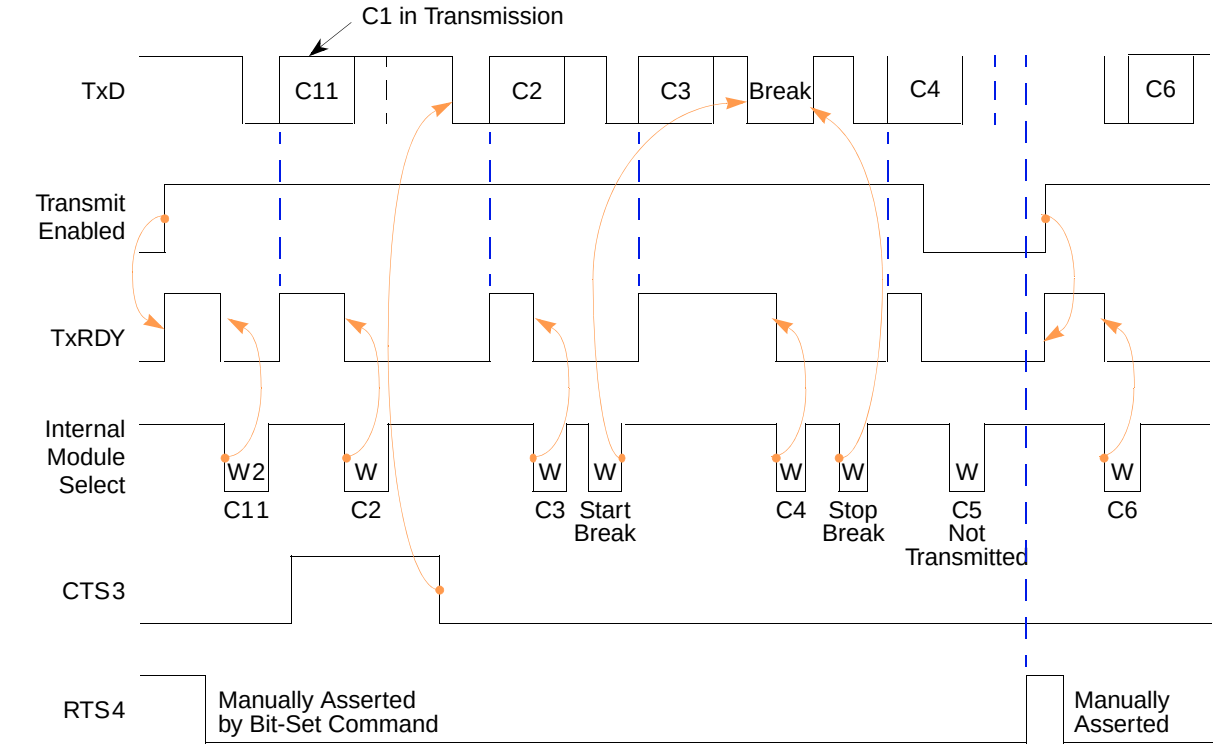
- The programmed number of data bits
- An optional parity bit
- The programmed number of stop bits

The LSB is sent first. Data is shifted from the Tx output on the falling edge of the clock source.

After the stop bits are sent, if no new character is in the Tx holding register, the TxD output remains high (mark condition) and the Tx empty bit, SR[TxEMP], is set. Transmission resumes and TxEMP is cleared when the CPU loads a new character into the PSC Tx buffer (TB).

- If the transmitter receives a disable command, it continues until any character in the Tx shift register is completely sent.
- If the transmitter is reset through a software command, operation stops immediately.
- If the clear-to-send operation is enabled, $\overline{\text{CTS}}$ must be asserted for the character to be transmitted.
- If $\overline{\text{CTS}}$ is negated in the middle of a transmission, the character in the shift register is sent and TxD remains in mark state until $\overline{\text{CTS}}$ is reasserted.
- If the transmitter is forced to send a continuous low condition by issuing a send break command, the transmitter ignores the state of $\overline{\text{CTS}}$.
- If the transmitter is programmed to automatically negate $\overline{\text{RTS}}$ when a message transmission completes, $\overline{\text{RTS}}$ must be asserted manually before a message is sent. In applications in which the transmitter is disabled after transmission is complete and $\overline{\text{RTS}}$ is appropriately programmed, $\overline{\text{RTS}}$ is negated one bit-time after the character in the shift register is completely transmitted. The transmitter must be manually re-enabled by reasserting $\overline{\text{RTS}}$ before the next message is sent.

Figure 25-42 shows the transmitter functional timing information.



NOTES:
1. Cn = transmit characters
2. W = write
3. MR2[TxCTS] = 1
4. MR2[TxRTvS] = 1

Figure 25-42. Timing Diagram—Transmitter

25.5.1.4 Receiving in UART Mode

After a hardware reset, all PSCs are in UART mode. The receiver is enabled through its CR, as described in Section 25.4.1.5, “Command Register (CR).” Figure 25-43 shows the receiver functional timing.

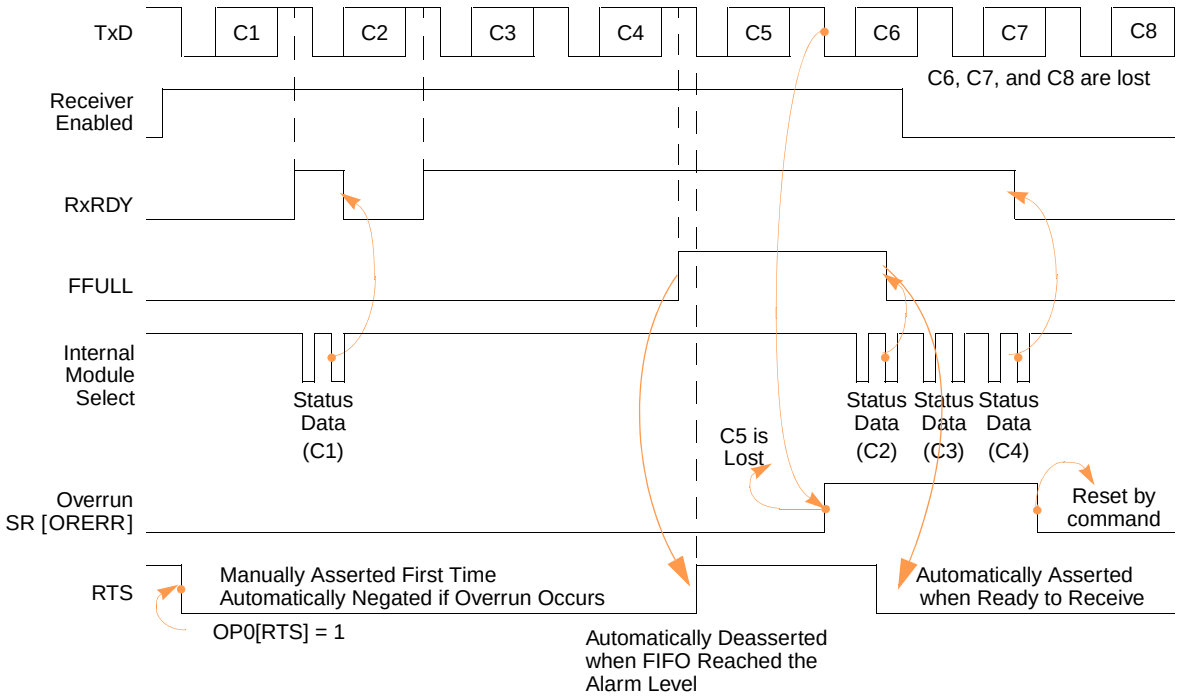


Figure 25-43. Timing Diagram—Receiver

When the receiver detects a high-to-low (mark-to-space) transition of the start bit on RxD, the state of RxD is sampled. It samples each 16× clock for eight clocks, starting one-half clock after the transition (asynchronous operation) or at the next rising edge of the bit-time clock (synchronous operation).

- If RxD is sampled high, start bit is invalid; a valid start bit search begins again.
- If RxD remains low, a valid start bit is assumed and receiver continues sampling input at 1-bit time intervals at the theoretical center of the bit. This continues until the proper number of data bits and parity, if any, is assembled and one stop bit is detected.

RxD input data is sampled on the rising edge of the programmed clock source. The LSB is received first. Data is then transferred to a receiver holding register and RxRDY bit is set. If the character is less than eightbits, the most significant unused bits in the receiver holding register are cleared.

If the MR1[RxRTS] bit was set to one, control the $\overline{\text{RTS}}$ line by writing to the output port register. For all user generated commands to the UART receiver, like enable RX, disable RX, set break or read data from the RX FIFO, set the associated $\overline{\text{RTS}}$ signal by writing the OP0 or OP1 register. However, the UART receiver automatically deasserts the $\overline{\text{RTS}}$ signal if the RX FIFO is full.

After the stop bit is detected, the receiver immediately looks for the next start bit.

- If a non-zero character is received without a stop bit (framing error) and RxD remains low for one-half of bit period after stop bit is sampled, the receiver operates as if a new start bit were detected. Parity error (PE), framing error (FE), overrun error (ORERR), and received break (RB) conditions set respective error and break flags in SR at the received character boundary and are valid only if RxRDY is set.

- If a break condition is detected (RxD is low for the entire character including the stop bit), a character of all zeros is loaded into the receiver shift register and SR[RB] is set. RxD must return to a high condition for at least one-half bit-time before a search for the next start bit begins.

The receiver detects the beginning of a break in the middle of a character, if the break persists through the next character time.

- If the break begins in the middle of a character, the receiver places the damaged character in the Rx FIFO stack and sets the corresponding SR error bit.
- If the break lasts until the next character time, the receiver places an all zero character into the Rx FIFO and sets SR[RB].

25.5.1.5 TimeOut Counter Behavior

The TimeOut counter is available during UART mode only. If this behavior is enabled then an internal counter can generate an interrupt if the time after the last received data word is bigger then the programmed limit. The first received data word after enabling this behavior will start the counter. A receive data word before the counter reach the limit will clear the counter value and starts the counter again. The value in the register defines the time base on number of clock events for the programmed UART clock source (32 or 10 clock events per bit depend on the selected prescaler).

25.5.1.6 Configuration Sequence for UART Mode

Table 25-28 shows the configuration sequences. This list includes the UART mode related registers only, not the other configure values like interrupt and FIFO configurations. PSC module registers can be accessed by word or byte operations.

Table 25-28. General Configuration Sequence for UART Mode

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
CSR	0xDD00	Select the clock source
SICR	0x0000_0000 or 0x0800_0000	Select the UART mode
MR1	0xXX	Select error mode, parity mode, and the parity type
MR2	0xXX	Select channel mode, port control, and stop-bit length
CTUR	0x00	Set the calculated baudrate depend on the IP bus clock
CTLR	0xD7	
IMR	0xXXXX	Select the desired interrupt
CR	0x05	Enable Tx and Rx

25.5.2 PSC in Codec Mode

After reset, all PSCs are in UART mode. The PSCs can be applied to one of the codec modes by writing the appropriate value to the SICR register. The other values should be initialized at the same time. During codec mode, the PSC can connect to codec interfaces with 8, 12,16, 20, 24, or 32 bit data. For all these modes, the PSC can be programmed to behave as a normal soft modem interface, SPI, ESAI, or I2S interface. The PSC codec supports all these modes the master mode (PSC drive the BCLK and FrameSync signal) or slave mode (PSC receive the BCLK and FrameSync signals) functionality. Independently from the mode (master or slave), the PSC can provide a MCLK (master clock) for an external codec device. This behavior eliminates the need for an external crystal for the external codec device. Figure 25-44 shows a simplified block diagram for the PSC codec mode. The important register to configure the PSC for codec mode are:

- SICR register—select the codec mode
- For master mode:
  - — select and enable MCLK divider
  - — CCR—select BCLK and FrameSync frequency
  - — CTUR—select FrameSync width
- CR register—enable or disable receiver and transmitter
- MR2 register

NOTE

The interface to the FIFO controller (FIFOC) supports 8, 16, and 32 bit access only. The data must be right-aligned in the FIFO, as shown in Table 25-29.

Table 25-29. FIFO Interface Support

Code mode	Access to the FIFO	used data
8-bit	8-bit	all 8-bit
12-bit	16-bit	12-bit, right assigned
16-bit	16-bit	all 16-bit
20-bit	32-bit	20-bit, right assigned
24-bit	32-bit	24-bit, right assigned
32-bit	32-bit	all 32-bit

25.5.2.1 Block Diagram and Signal Definition for Codec Mode

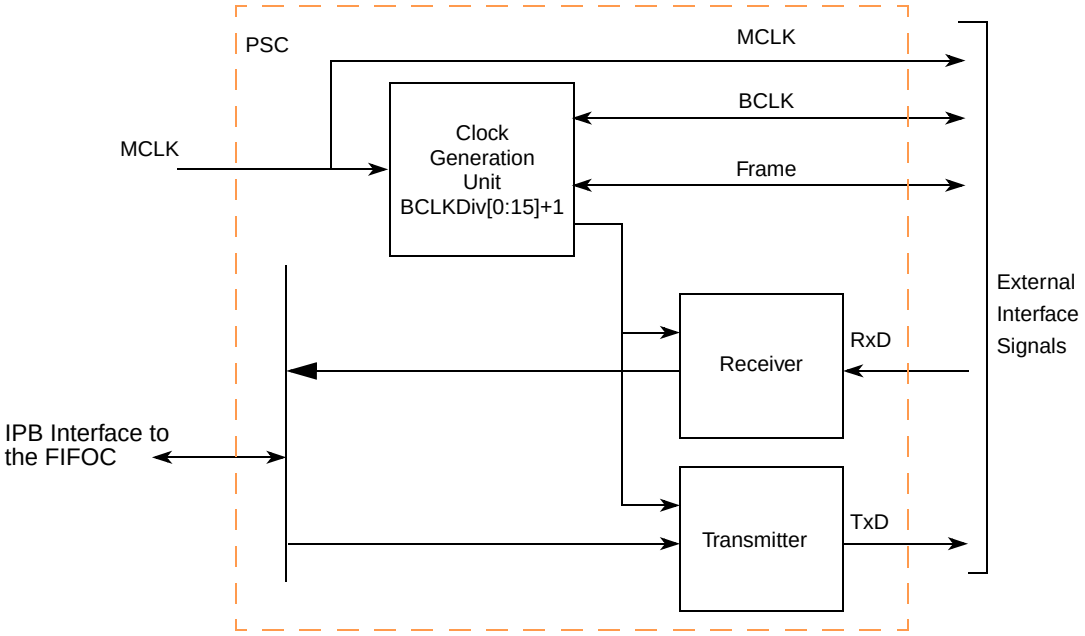


Figure 25-44. PSC Codec Block Diagram

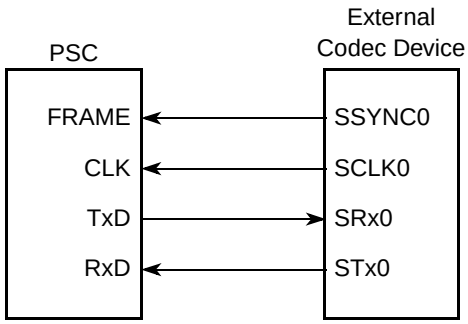


Figure 25-45. PSC Codec Interface in Slave Mode

Table 25-30. PSC Signal Description for Codec Mode

Signal	Description
TxD	Transmitter Serial Data Output. Data is shifted out on TxD on the falling or rising edge of the clock source. Transfers can be specified as LSB or MSB first. TxD is held low when Tx is disabled or idle. • Data shifted out on the rising edge of CLK if SICR[ClkPol] equals 0 • Data shifted out on the falling edge of CLK if SICR[ClkPol] equals 1 • Data send MSB first if SICR[SHDIR] equals 0 • Data send LSB first if SICR[SHDIR] equals 1

Table 25-30. PSC Signal Description for Codec Mode (continued)

Signal	Description
RxD	Receiver Serial Data Input. Data received on RxD is sampled on the falling or rising edge of the clock signal. Transfers can be specified as either LSB or MSB first. • Data sampled on the rising edge of CLK if SICR[ClkPol] equals 1 • Data sampled on the falling edge of CLK if SICR[ClkPol] equals 0 • Data sampled MSB first if SICR[SHDIR] equals 0 • Data sampled LSB first if SICR[SHDIR] equals 1
Frame	Frame Sync. In codec mode Frame can be driven from an external Codec or can be generate by the internal clock logic. Frame can be programmed as active High or active Low. • The frame sync input from the external Codec if SICR[GenClk] equals 0 • The frame sync output to the external Codec if SICR[GenClk] equals 1 • Frame sync is active low if SICR[SyncPol] equals 0 • Frame sync is active high if SICR[SyncPol] equals 1
CLK	Bit Clock. In codec mode CLK is: • The clock input from the external Codec if SICR[GenClk] equals 0 • The clock output to the external Codec if SICR[GenClk] equals 1
MCLK	Clock output for an external codec

25.5.2.2 Codec Clock and FrameSync Generation

The serial BCLK and the FrameSync can be inputs that come from an external codec device or they can be internally generated by the PSC and provided as outputs to the external device, under control of the SICR[GenClk] bit. When the SICR[GenClk] bit = 0, the BCLK and the FrameSync are inputs. In this case, the FrameSync width can be anything from one BCLK period up to the total FrameSync length/period minus one BCLK. If the GenClk bit = 1, the PSC generates the BCLK and the FrameSync signal. Figure 25-46 shows how the PSC generates the clocks.

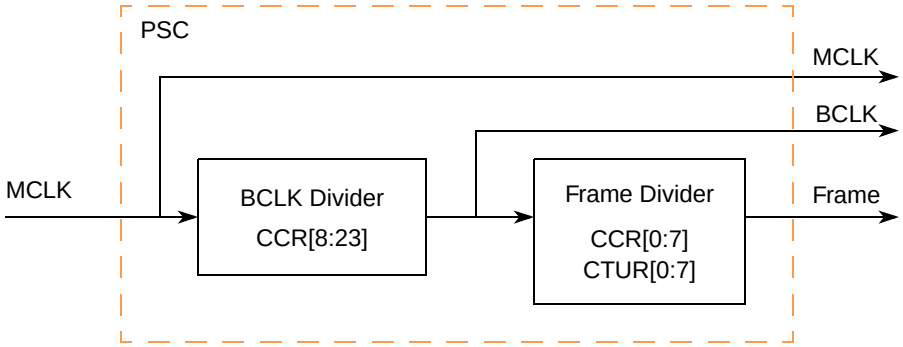


Figure 25-46. Clock Generation Diagram for Codec Mode

The source for the internal clock generation is the MCLK. The PSC provides the clock to the external codec divided independently whether the PSC configured as a master (provides BCLK and FrameSync) or as a slave (receives the clock signals).

Each PSC consists of a CCR register to generate a BCLK and a FrameSync signal. If the PSC is configured as a master and the MCLK is available, the PSC generates both clock signals independently of whether the

transmitter or receiver is enabled or not, as opposed to the SPI behavior. Equation 25-3 and Equation 25-4 show the calculation.

Eqn. 25-3

$$BCLK = \frac{MCLK}{BCLKDiv[0:15] + 1}$$

Eqn. 25-4

$$FrameSync\ Length = Frame\ SyncDiv[0:7] + 1$$

When the FrameSync is an output, the CTUR register can program the pulse width. This register defines the number of BCLK cycle during the FrameSync signal is active. The default reset value for this register is 0x00. Therefore, the default FrameSync width is one BCLK. See Equation 25-5.

Eqn. 25-5

$$Frame\ sync\ width = CTUR[0:7] + 1$$

25.5.2.3 Transmitting and Receiving in Soft Modem Codec Mode

The PSC supports the full duplex soft modem mode, data is received and transmitted at the same time. To start the full duplex transmission, the Tx and the Rx must be enabled by writing the according value to the CR register. It's also possible to only use the receiver. For this case, only the Rx enable bit in the CR register must be set to one. However, it's not possible to use the transmitter without the receiver. To transmit data only, the receiver must be enabled. The received data and the according status and interrupt bits can be ignored.

If the receiver is enabled, the PSC samples data from the receive line after detecting the start of frame condition. The receiver converts the serial data from the RX line to parallel data words and writes the data to the RxFIFO. The data word length depends on the programmed word length. If no data exists on the Rx line, the receiver writes zeros to the RxFIFO until the data word width was reached. The receiver waits until the next start of frame condition is detected. The transmitter converts the parallel data from the TxFIFO to a serial data stream on the TX line. If the TxFIFO is empty during the transmit state, the Tx line is zero. If the last bit of the data word is sent, the transmitter waits until the next start of frame condition is detected.

When SICR[GenClk] = 1, the PSC is in master mode and generates the BCLK and the FrameSync signal from the internal clock system, as described in Section 25.5.2.2, "Codec Clock and FrameSync Generation."

Figure 25-47 shows a codec interface diagram example for soft modem master mode. The different parameters to define the interface are follows:

- Frame Sync Polarity SICR[SyncPol]. The leading edge is defined as a rising edge if bit SICR[SyncPol] equals 1 or a falling edge if SICR[SyncPol] equals 0.

- BCLK Polarity SICR[ClkcPol]. When bit SICR[ClkPol] equals 0, data is shifted out on the rising edge of bit clock and sampled on the falling edge of BCLK. Otherwise, data is shifted out on the falling edge and sampled on rising edge of bit clock.
- FrameSync width CTUR. Defines the number of BCLK while the FrameSync is active.
- FrameSync length CCR[FrameSyncDiv]. Defines the number of BCLK until the next frame starts.
- Data length SICR[SIM]. Defines the data with the receive and transmit data, 8-, 12-, 16-, 20-, 24-, or 32-bit per word are possible. In codec 20 and 24 mode, each data sample uses an entire 32-bit longword in the Tx FIFO. The least significant (right-hand) byte is not used. Data should be written to the Tx FIFO four bytes at a time.
- Delay of time slot 1 SICR[DTS1]. The PSC starts to send a sample at the leading edge of FrameSync SICR[DTS1] if it equals 0 or 1 bit-clock cycle after the leading edge of FrameSync SICR[DTS1] if it equals 1.
- Data shift direction SICR[SHDIR]. Data shifted out LSB first if SICR[SHDIR] equals 1. Otherwise, data shifts out MSB first if SICR[SHDIR] equals 0.

In the codec soft modem mode, the PSC sends only one data word per frame.

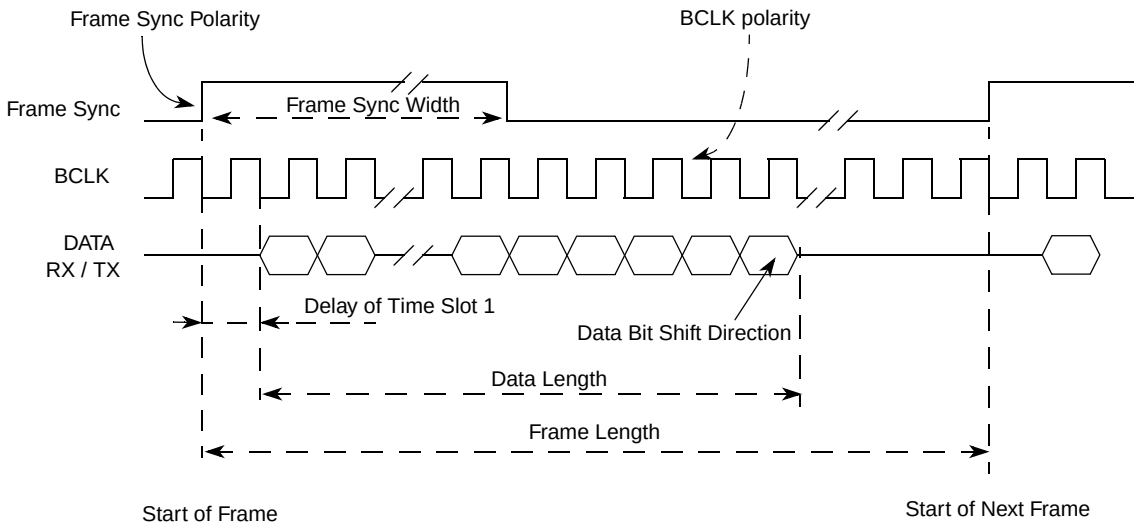


Figure 25-47. Soft Modem Codec Interface Diagram

Table 25-31 shows an example how to configure PSC1 as:

- PSC in slave mode
- 16-bit soft modem mode
- Data is sampled on the falling edge of BCLK
- FrameSync is low true
- MSB first, transfer starts with leading edge of FrameSync

Table 25-31. 16-Bit Soft Modem Slave Mode

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x0210_0000	Select the 16 bit Codec mode, MSB first, DTS1 = 0, slave mode
CR	0x05	Enable Tx and Rx

Table 25-32 shows an example of how to configure the PSC2 as:

- PSC in Master mode
- 32-bit soft modem mode
- Data is sampled on the rising edge of BCLK
- FrameSync is low true
- LSB first, transfer starts one cycle after the leading edge of FrameSync
- Set MCLK frequency to 33 MHz
- Set BCLK frequency to 250 kHz
- FrameSync every 35 BCLK
- Set FrameSync width to 3 BCLK

Table 25-32. 32-Bit “Soft Modem“Master Mode

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x3FA0_0000	Select the 32bit Codec mode, LSB first, DTS1 = 1, master mode
CCR	0x2283_0000	select the BCLK and FrameSync frequency
CTUR	0x02	select the FrameSync width
CR	0x05	Enable Tx and Rx

25.5.2.4 Transmitting and Receiving in ESAI Mode (Enhanced Serial Audio Interface)

The ESAI transmission is similar to the soft modem mode. Therefore, the configuration as described in Section 25.5.2.3, “Transmitting and Receiving in Soft Modem Codec Mode.” The difference is that the ESAI protocol allows to transmit and receive more than one data word per frame. To enable the ESAI mode, the SICR[ESAI] bit must be set. The PSC calculates how many data words the transmitter sends and how much data the receiver expects. Figure 25-48 shows the ESAI transmission diagram.

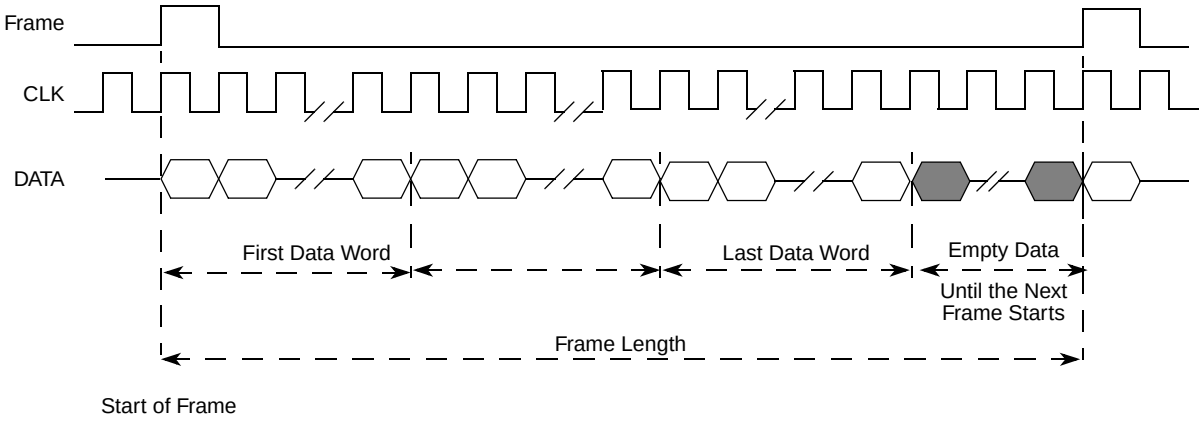


Figure 25-48. ESAI Data Transmission

Table 25-33 shows an example how to configure the PSC1 as ESAI master. For the slave mode, the bit SICR[GenClk] must be cleared and the configuration of the CCR register can be ignored. In this configuration example, the PSC sends three data words with 16-bit data in the 52 BCLK frame length. The last 4 bits in the frame are empty (0).

- Use PSC1 as ESAI master
- 16-bit data, LSB first
- BCLK frequency 4 MHz
- FrameSync length 52 bit
- Data shifted out on the rising edge of BCLK
- Data transfer starts on FrameSync is active
- FrameSync is active high

Table 25-33. 16-Bit ESAI Master Mode for PSC1

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x12D2_0000	Select the 16-bit Codec ESAI master mode, LSB first, DTS1 = 0
CCR	0x3303_0000	Set the FrameSync length (52 bit) and SCKL frequency
CR	0x05	Enable Tx and Rx

25.5.2.5 Transmitting and Receiving in I2S Master Mode

The I2S transmission is similar to the soft modem mode. Therefore, the configuration is as described in Section 25.5.2.3, “Transmitting and Receiving in Soft Modem Codec Mode.” The difference is that during the I2S word transmission the FrameSync signal (LRCK) is stable for the complete data word and is the opposite for the next one. To enable the I2S mode, the SICR[I2S] bit must be set. The SICR[SyncPol] bit defines if the frame starts with a low LRCK signal or with a high LRCK signal. If the transmitter detects the start condition, it starts to send the data from the TxFIFO. If the receiver detects a start condition, it starts to write the data from the RX line to the RxFIFO. The FIFO does not provide the ability to mark the

data in the FIFO. Therefore, only the order in the FIFO defines whether the data was received or transmitted during the high or low phase of the LRCK. Figure 25-49 shows the I2S transmission diagram.

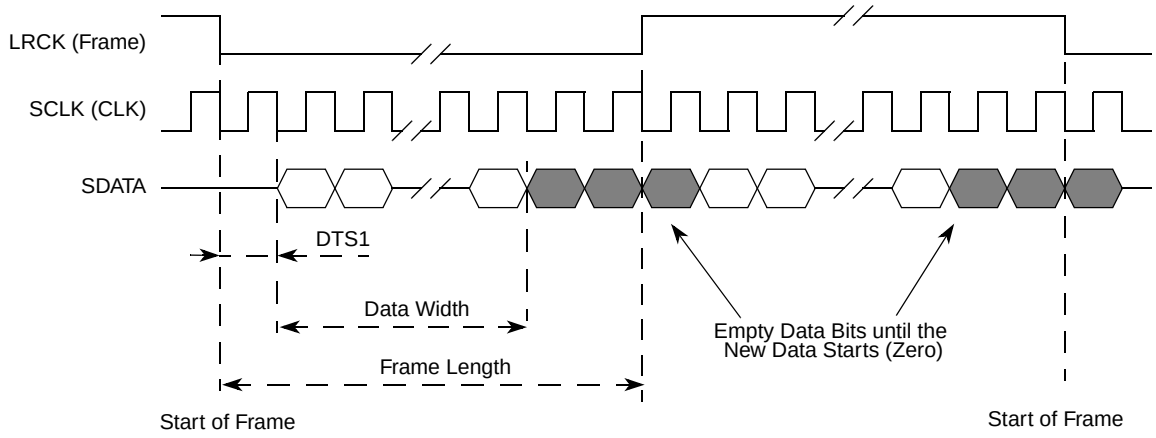


Figure 25-49. I2S-Data Transmission

Table 25-34 shows an example of how to configure the PSC1 as I2S master. For the slave mode the bit SICR[GenClk] must be cleared and the configuration of the CCR register can be ignored.

- Use PSC1 as I2S master
- 32-bit data, MSB first
- SCLK frequency 1 MHz
- FrameSync width 40 bits
- Data shifted out on the falling edge of SCLK
- Data transfer starts one CLK cycle after the FrameSync is active
- Frame starts with LRCK low

Table 25-34. 32-bit I2S Master Mode for PSC1

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x2FE0_0000	Select the 32-bit Codec I2S master mode, MSB first, DTS1 = 1
CCR	0x270F_0000	Set the FrameSync width (40 bit) and SCKL frequency
CR	0x05	Enable Tx and Rx

25.5.2.6 Transmitting and Receiving in SPI Mode

The PSC supports a full duplex SPI interface. This mode is chosen by setting SICR[SPI] equal to 1, which must be true for the MSTR, CPOL, CPHA, and UseEOF bits in the SICR register to take effect. In SPI mode, the SICR[SIM] bits must also be set to select the data width. To configure the PSC to act like an SPI master, set SICR[MSTR] equal to 1 or set SICR[MSTR] equal to 0 to configure the PSC as an SPI slave. When the SICR[MSTR] bit is set, SICR[GenClk] must also be set to 1 because the PSC is driving the SPI clock line. When SICR[MSTR] equals 0, SICR[GenClk] must be set to 0 because the external SPI is driving the SCK clock line. The CPOL and CPHA bits in the SICR register operate exactly the same

way as they do in an SPI, and their values must be the same as the CPOL and CPHA bits in the SPI device communicating with the PSC. The SICR[UseEOF] bit has an effect only when SICR[MSTR] equals 1 for master mode. If the UseEOF bit is cleared, only one data word (8, 12, 16, or 32 bit width depending on the SICR[SIM] field) sent before slave select ($\overline{SS}$) goes high/inactive.

When SICR[UseEOF] equals 1, the number of bytes transferred prior to $\overline{SS}$ going high is controlled by the EOF flag inside the Tx FIFO.

As the PSC reads bytes out of the Tx FIFO, it holds $\overline{SS}$ low/active until it transmits a byte whose EOF flag is set. In this mode, there is virtually no limit on how many bytes can be sent in one SPI transfer. EOF mode is supported to enable the user to transfer an increased quantity of data in one frame. When using EOF mode, the PSC reads bytes out of the Tx FIFO, the $\overline{SS}$ signal is driven low (asserted), and the SPI continues to transmit. The last data of the frame is sent after the EOF flag is set. This presence of the EOF flag shows that the last data in the frame has been reached. EOF mode should not be used for single data transfers, and is limited by data packages that must be bigger than the chosen codec size. When using EOF mode, the next data EOF frame flag should not be used to toggle the $\overline{SS}$ signal after every individual transfer when in EOF mode. If the SPI is used to transmit data in single codec size packages, the UseEOF bit in the SICR register should be cleared (0), and the value of the SIM field in this register should be set for the appropriate data transfer size.

To mark a data word with the EOF flag during a IPB transfer, set the IRCR2[NXTEOF] bit before writing the last data word to the TX FIFO. This bit is cleared after the next write access to the TX FIFO.

The SICR[SHDIR] bit controls the shift direction in SPI mode, as it does in the non-SPI codec modes. The DTS1, ESAI, ClkPol, SyncPol, CellSlave, and Cell2xClk bits in the SICR register have no effect in SPI mode.

In SPI master mode, the BCLK (SCK) frequency is generated by dividing down the MCLK frequency; see Section 25.5.2.2, “Codec Clock and FrameSync Generation.” In addition to the BCLK generation, the DSCLK delay and the DTL delay must be defined. The DSCLK defines the delay between the $\overline{SS}$ going active and the first BCLK (SCK) clock pulse transition. The DSCLK delay is created by dividing down the MCLK frequency. The delay between consecutive transfers is created by dividing down the IPB clock frequency. For more information about the delay generation, see the description of the CTUR, CTLR and CCR registers.

Eqn. 25-6

$$\text{DSCKL delay} = \frac{\text{CCR[0:7]} + 1}{\text{MCLK}}$$

$$\text{DTL} = \frac{\text{CT[0:15]} + 2}{\text{IPB clock frequency}} + \frac{3}{\text{MCLK frequency}}$$

where:
CT[0:15] = {CTUR[0:7], CTLR[0:7]}

In SPI master mode the PSC controls the serial data transfers. If the Tx FIFO becomes empty (underrun) or the Rx FIFO becomes full (overflow) in the middle of a multi-byte transfer, rather than set the Tx underrun or Rx overflow status bits, the PSC keeps the slave select signal low/active and stops the SCK

serial clock. As long as the RX FIFO becomes not full and the TX FIFO becomes not empty, the transfer is ongoing.

In SPI slave mode, the MCLK must be running/enabled even though it is not used to generate the serial clock SCK, which is provided by the external master SPI device. The frequency of MCLK is not critical, as long as it is faster than the SCK frequency.

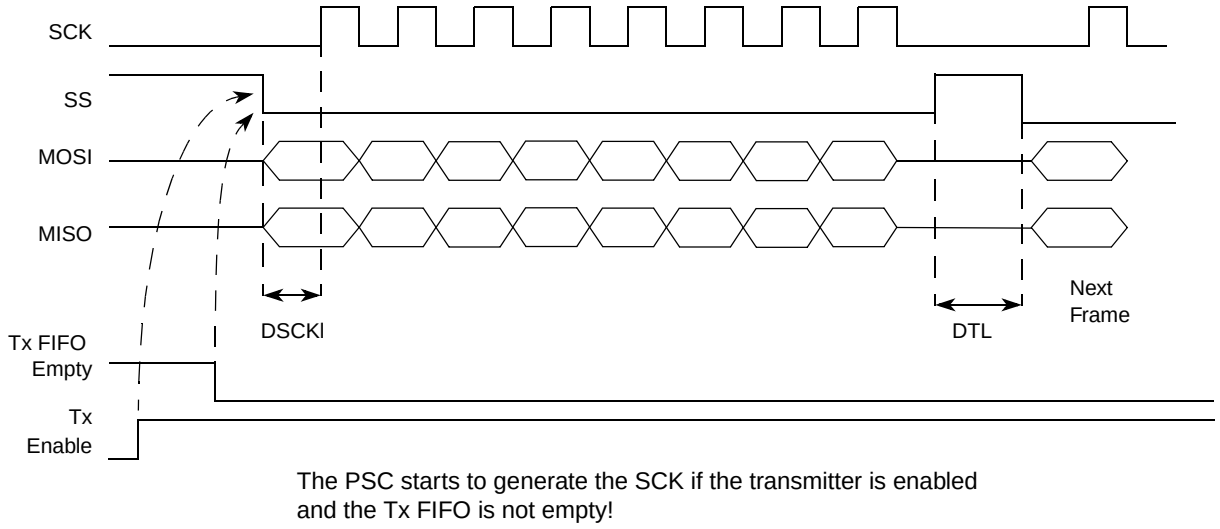


Figure 25-50. SPI Parameter

Table 25-35 shows an example of how to configure the PSC3 as SPI master.

- 32-bit data
- Clock is active high, CPOL = 0
- The first SCK edge is issued one half cycle into the data transfer; CPHA = 0
- MSB first
- Baud rate 1 Mbit/s
- DSCLK delay = 0.5 µs
- DTL delay = 2.0 µs

Table 25-35. 32-bit SPI Master Mode for PSC3

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x0F00_C000	Select the 32bit Codec SPI master mode, MSB first, CPOL = 0,CPHA = 0
CCR	0x070F	Set the SCK and DSCKL delay
CTUR	0x00	Set the DTL delay 2us
CTLR	0x84	
CR	0x05	Enable Tx and Rx

Table 25-36 shows an example of how to configure the PSC2 as SPI slave.

- Use PSC2 as SPI slave
- 8-bit data
- Clock is active low, CPOL = 0
- The first SCK edge is issued at the beginning of the data transfer; CPHA = 1
- MSB first

Table 25-36. 8-bit SPI Slave Mode for PSC2

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x0100_9000	Select the 8bit Codec SPI slave mode, MSB first, CPOL = 0; CPHA = 1
CR	0x05	Enable Tx and Rx

25.5.3 PSC in AC97 Mode

After reset, all PSCs are in UART mode. AC97 mode is chosen by setting the SICR[SIM] equal to 0x3. The other SICR field should be initialized at the same time. The important registers to configure the PSC for AC97 mode are:

- SICR register—select the codec mode
- CR register—enable or disable receiver and transmitter
- OP0, OP1 register—generate the reset pulse for the external device

25.5.3.1 Block Diagram and Signal Definition for AC97 Mode

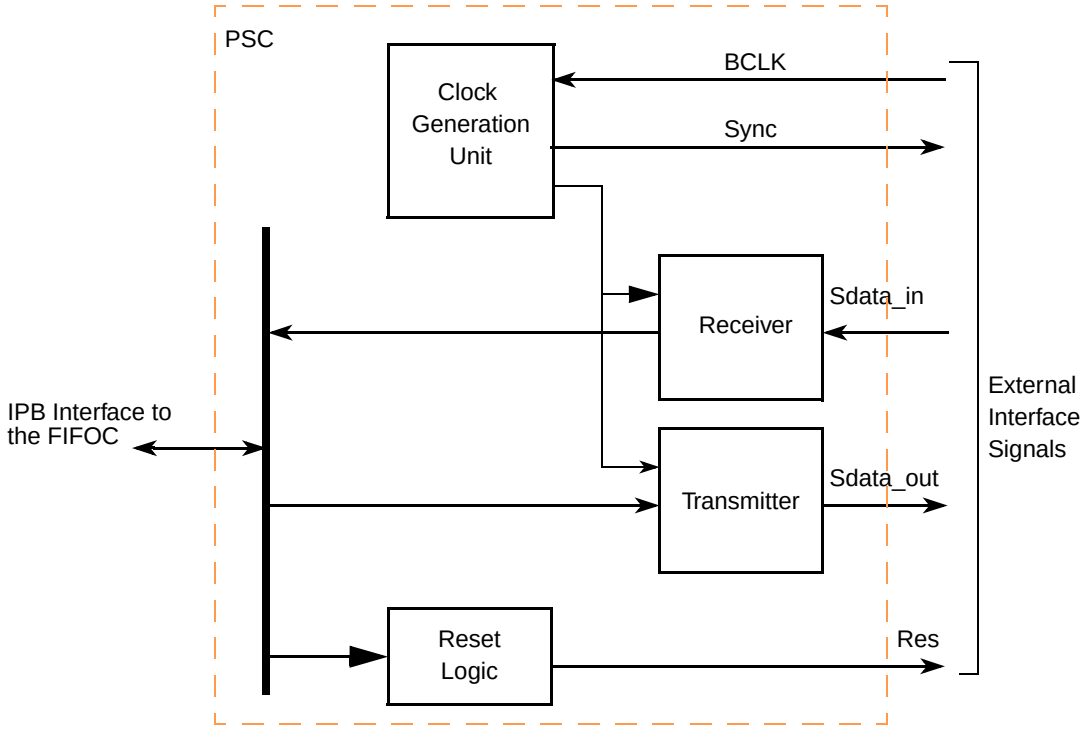


Figure 25-51. PSC AC97 Block Diagram

Figure 25-51 shows the simplified PSC Block Diagram for AC97 mode. The BCLK is an input from the external codec. The PSC divides BCLK by 256 to generate a Frame pulse (Sync) that is high for 16 BCLK cycles. The PSC can only work as and AC97 controller. This means the PSC receives the BCLK from the external AC97 codec and provides the associated frame signal. In AC97 mode, the clock and frame relations are fixed. Therefore, the CCR register and the SICR[GenClk] bit are not used. Table 25-37 shows the pin definition for the AC97 mode, and Figure 25-52 shows an AC97 interface. A general-purpose I/O (GPIO) is used as a reset to the external AC97 device.

Table 25-37. PSC Signal Description for AC97 Mode

Signal	Description
Sdata_out	Transmitter Serial Data Output. Data is shifted out on TxD on the rising edge of the clock signal. Transfers must be specified as MSB first.
Sdata_in	Receiver Serial Data Input. Data received on RxD is sampled on the falling edge of the clock signal. Transfers must be specified as MSB first.
Sync	In AC97 mode, Sync is the frame sync, or start-of-frame (SOF), output to the external AC97 Controller. In this mode, the AC97 BCLK, which is input on CLK, is divided by 256 to generate the Sync.
BCLK	BCLK. In AC97 mode, CLK must be driven by the serial bit-clock from the external AC97 Controller.
Res	Reset signal to the external AC97 device

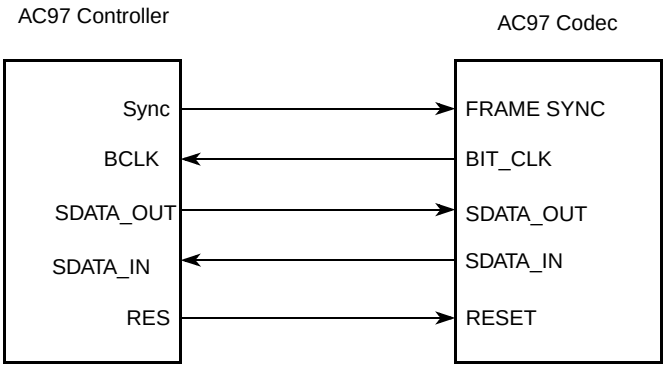


Figure 25-52. PSC — AC97 Interface

Figure 25-53 shows the timing diagram for the AC97 interface. For more AC97 Controller interface information, see the *Audio Codec'97 Component Specification.*

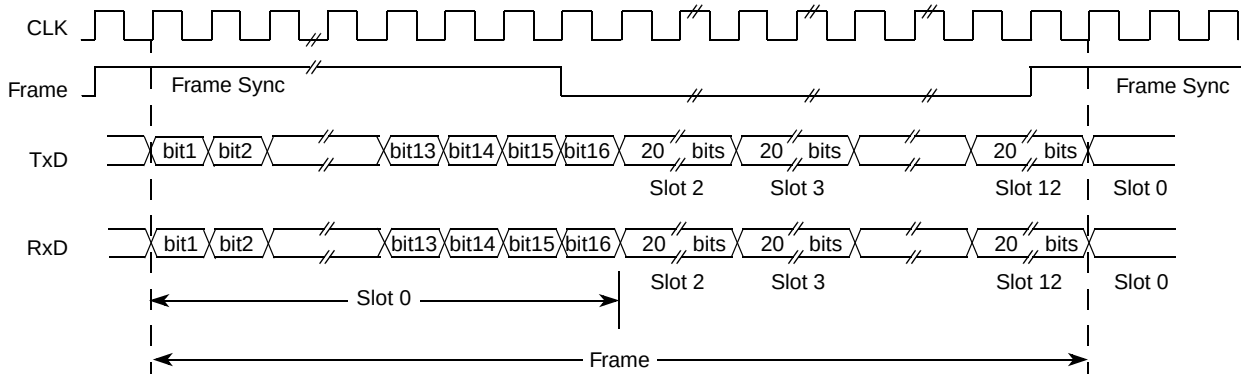


Figure 25-53. Timing Diagram—AC97 Interface

25.5.3.2 Generating a Reset Pulse for the External AC97 Codec Device

The following sequence generates a reset pulse for the external AC97 device:

1. $\overline{\text{Res}}$ line is high, after power up
2. Write 0x02 to the OP1 register; $\overline{\text{Res}}$ line goes low
3. Write 0x02 to the OP0 register; $\overline{\text{Res}}$ line goes high

NOTE

Some AC97 devices goes to a test mode if the sync line is high while the $\overline{\text{Res}}$ line is low (reset phase). To avoid this behavior, the sync line must also be forced to zero during the reset phase. To do that, the pin muxing should switch to GPIO mode and the GPIO control register should be used to control the output lines.

25.5.3.3 AC97 Low-Power Mode

A General-Purpose I/O (GPIO) must be used as an AC97 reset output pin. The PSC monitors the first three time slots of each Tx frame to detect the power-down condition for the AC97 digital interface. The power-down condition is detected as follows:

1. The first three bits of slot 1 must be set, indicating Tx frame and slots 1 and 2 are valid.
2. Slot 2 holds the power-down register (0x26) address in the external AC97 device.
3. Slot 3 has one in the fourth bit (bit 12/PR4 in power-down register 1), as defined in the AC97 specification.

Low-power mode can be left through a warm or cold reset. The CPU does a warm reset by setting SICR[AWR] for at least 1µs. This asserts the FRAME frame sync output in AC97 mode. The CPU does a cold reset in two steps:

1. Writes 0 to whichever GPIO is the active low AC97 reset pin for the minimum time specified in the AC97 specification.
2. Writes 0 to PSC1 or PSC2 SICR[ACRB]. CPU should set this bit after writing 1 to the GPIO used for the AC97 reset pin.

NOTE

Step 2 (above) is required so the PSC knows when an AC97 cold reset is occurring.

25.5.3.4 Transmitting and Receiving in AC97 Mode

The data transmission is the standard AC97 one, see Section 25.5.3.1, “Block Diagram and Signal Definition for AC97 Mode”. The AC97 controller is able to generate the time slot0,1 and 2 data on the transmit site and analyzes received time slot0,1, and 2. The used data slots (3 to 12) must be in the FIFOs.

The RX_SLOTS field in the AC97Slots register specify the expected RX data slots. If the received slots do not match this specification, the receiver ignores all data slots from the current frame and sets the SR[UNEX_RX_SLOT] bit. Only the expected and valid tagged data slots are in the RxFIFO. This functionality guarantees the software can assign the data in the RxFIFO to an AC97 slot. Only the order in the RxFIFO marks the AC97 slot number.

The TX_SLOTS field in the AC97Slots register defines which data slots are sent. All data for these slots must be in TxFIFO. The transmitter generates the related slot0 tag data. If the TxFIFO is empty, the transmitter tags the frame as empty. The transmitter sends data if the receiver detects the codec ready state for the current frame. the Tx FIFO contains the specified data words (defined in the AC97Slots register), and the slot request for the specified slots was active (slot request bit was zero in the previous frame). If the AC97 codec set a slot request to one, the transmitter sends a complete empty frame because the transmitter is cannot send a port of the required slots without changing the order of the data in the FIFO.

If the software sends a command to the AC97 codec, the control register index and the control register write data values must be written to the AC97CMD register. A write access to any word of this register triggers the transmitter to send out the register value, synchronous the SR[CMD_SEND] bit was set. The transmitter generates a slot0 tag that marks slot1 and slot2 as valid slot. If the receiver was able to send out the command data, the SR[CMD_SEND] bit is cleared.

If the receiver detect a valid data in time slot2, the SR[DATA_VALID] bit was set by the receiver. The software can read the received data from the AC97Data register, synchronous the read access to this register clears the SR[DATA_VALID] bit. If the receive detect an additional command data before the previous data was read out, the SR[DATA_OVR] bit was also set to one. The previous received command data word goes lost. A read access to the AC97Data register clears the SR[DATA_VALID] and SR[DATA_OVR] register. Table 25-38 shows an example how to configure the AC97 controller. In this example, the AC97 controller only sends time slot 3 and slot4 data and expects data for time slot9, 10, 11, and 12 on the receive site. For this purpose, the software must write two data words to the TxFIFO for one complete AC97 frame and read four data words from the RxFIFO per frame.

Table 25-38. General Configuration Example AC97 Mode

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x0301_0000	Select the enhanced AC97 mode
AC97Slots	0x0300_000F	Define the expected receive and transmit slots
IMR	0xXXXX	Select the desired interrupt
CR	0x05	Enable Tx and Rx

25.5.4 Local Loop-Back Mode

Figure 25-54 shows how TxD and RxD are internally connected in local loop-back mode. This mode is for testing the operation of a local PSC module channel by sending data to the transmitter and checking data assembled by the receiver to ensure proper operations.

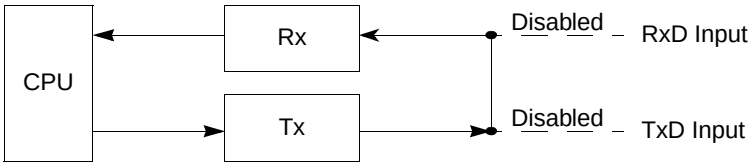


Figure 25-54. Local Loop-Back

Features of this local loop-back mode are:

- Transmitter and CPU-to-receiver communications continue normally.
- RxD input data is ignored.
- TxD data is held marking.
- The receiver is clocked by the transmitter clock.
- Transmitter must be enabled, but the receiver need not be enabled.

25.5.5 Remote Loop-Back Mode

In remote loop-back mode, shown in Figure 25-55, the channel automatically transmits received data bit-by-bit on the TxD output. The local CPU-to-transmitter link is disabled. This mode is useful in testing

receiver and transmitter operation of a remote channel. For this mode, the transmitter uses the receiver clock.

Because the receiver is not active, received data cannot be read by the CPU and error status conditions are inactive. Received parity is not checked and is not recalculated for transmission. Stop bits are sent as they are received. A received break is echoed as received until the next valid start bit is detected.

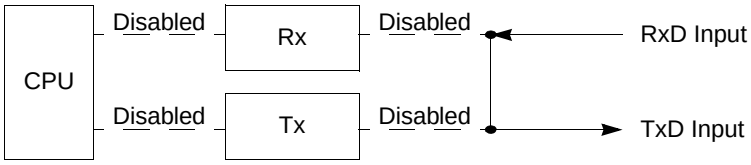


Figure 25-55. Remote Loop-Back

25.5.6 Multidrop Mode

Setting MR1[PM] programs the PSC to operate in a walk-up mode for multidrop or multiprocessor applications. In this mode, a master can transmit an address character followed by a block of data characters targeted for one of as many as 256 slave stations.

Although slave stations have their channel receivers disabled, they continuously monitor the masters data stream. When the master sends an address character, the slave receiver channel notifies its respective CPU by setting SR and generating an interrupt (if programmed to do so). Each slave station CPU then compares the received address to its station address and enables its receiver if it wishes to receive the subsequent data characters or block of data from the master station. Slave stations not addressed continue monitoring the data stream. Data fields in the data stream are separated by an address character. After a slave receives a block of data, its CPU disables the receiver and repeats the process.

Figure 25-56 shows functional timing information for multidrop mode.

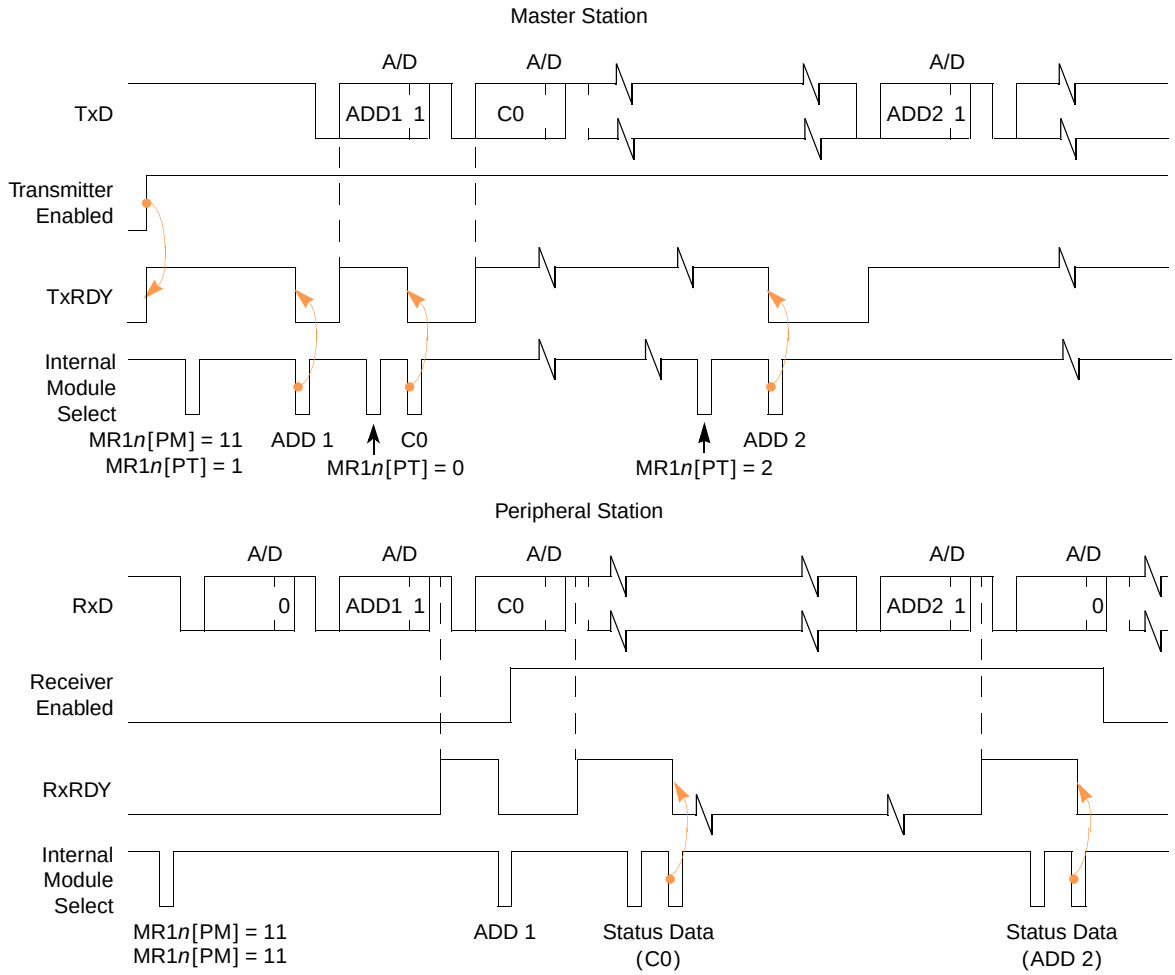


Figure 25-56. Timing Diagram—Multidrop Mode

A character sent from the master station consists of:

- A start bit
- A programmed number of data bits
- An address/data (A/D) bit flag
  - A/D=1 indicates an address character
  - A/D=0 indicates a data character
- A programmed number of stop bits

A/D polarity is selected through MR1[PT]. MR1 should be programmed before enabling the transmitter and loading the corresponding data bits into the Tx buffer.

In multidrop mode, the receiver continuously monitors the received data stream, regardless of whether it is enabled or disabled.

- If the receiver is disabled, it sets the RxRDY bit and loads the character into the receiver holding register FIFO stack, provided the received A/D bit is 1 (address tag). If the received A/D bit is 0 (data tag), the character is discarded.

- If the receiver is enabled, all received characters are transferred to the CPU through the receiver holding register stack during read operations.

In either case, data bits are loaded into the data portion of the stack while the A/D bit is loaded into the status portion of the stack normally used for a parity error (SR[PE]).

Framing error, overrun error, and break detection operate normally. The A/D bit takes the place of the parity bit. Parity is neither calculated nor checked. Messages in this mode may continues to contain error detection and correction information. One way to provide error detection if 8-bit characters are not required is to use software to calculate parity and append it to the 5-, 6-, or 7-bit character.

Chapter 26
PSC Centralized FIFO Controller (FIFOC)

26.1 Introduction

The MPC5125 has a centralized FIFO controller that contains data to be transmitted and received data for all 10 PSC modules. For each PSC module, one TX and one RX FIFO space is available. The size of each memory slice is programmable. If the unused FIFO slices are set to zero, another FIFO slice can use this area. To maintain data consistency, multiple slices cannot share the same space. The available memory space for all slices together is 32 × 1024 (4 KB).

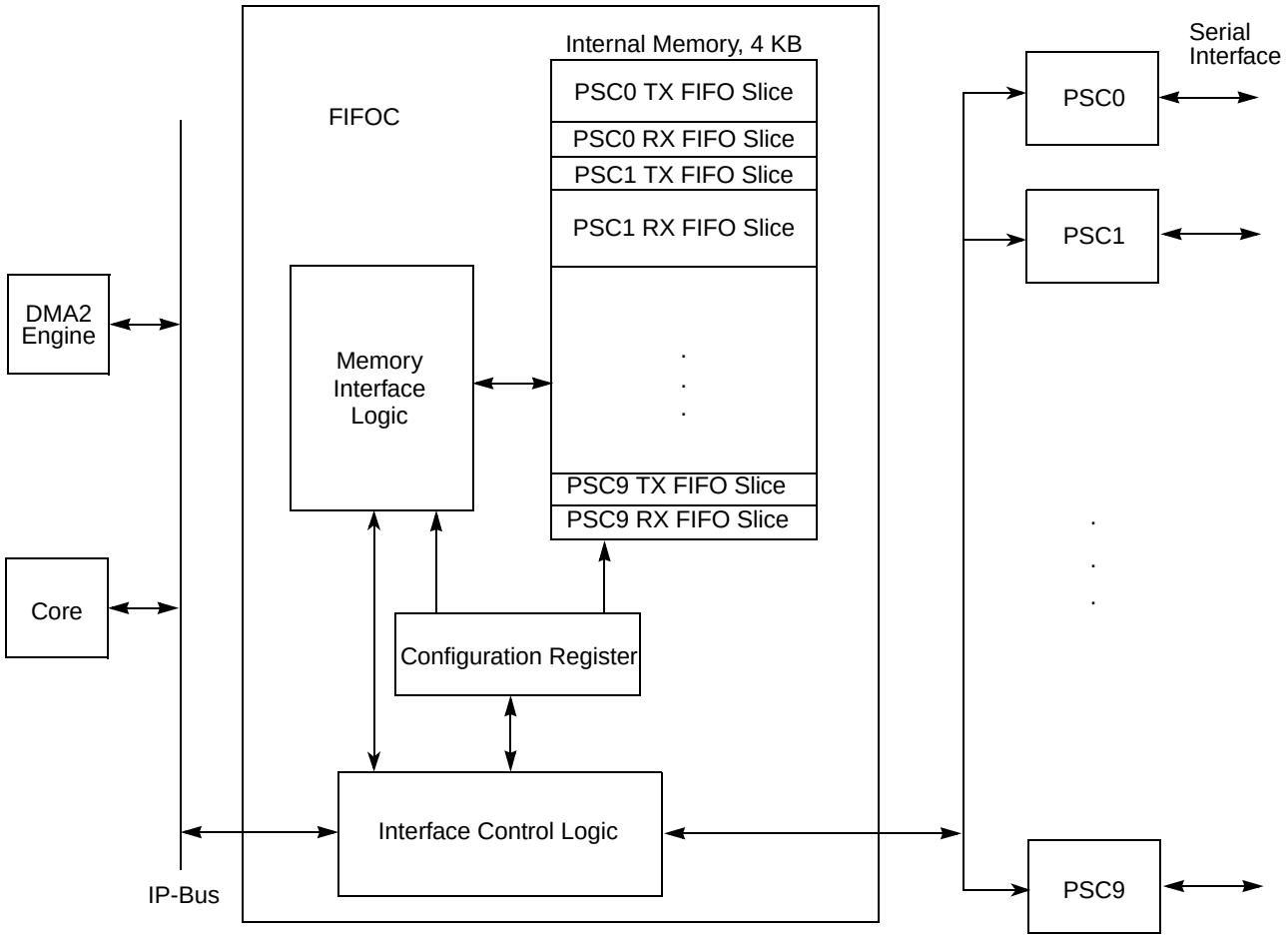


Figure 26-1. FIFOC Overview

26.1.1 Features

Features of the FIFO controller are:

- Independent programmable memory size for transmitter and receiver for each PSC
- All PSCs share the FIFO memory; each PSC FIFO can use as much memory as available depending on the configuration
- Dynamic clock gating function to save power during normal operation
- Independent programmable request signals to the core and DMA2
- Debug mode to overwrite internal address pointer and write access to ISR register to generate interrupts independent from the interrupt status

26.1.2 Modes of Operation

Behind the normal FIFO controller operation, the FIFOC provides a debug mode that allows the software to manipulate the internal generated pointer register.

26.2 Memory Map and Register Definition

Table 26-2 provides a simplified memory map for the PSC and FIFOC. Table 26-2 shows the registers for the PSCs. Table 26-3 shows the registers for the FIFO controller.

Table 26-1. PSC and FICOC Memory Map

Address	Register area
0xFF41_1000 (PSC0_BASE)	0x00–0x54: PSC0 registers
	0x58–0x7F: Reserved
	0x80–0xFF: PSC0 FIFOC registers
0xFF41_1100 (PSC1_BASE)	0x00–0x54: PSC1 registers
	0x58–0x7F: Reserved
	0x80–0xFF: PSC1 FIFOC registers
0xFF41_1200 (PSC2_BASE)	0x00–0x54: PSC2 registers
	0x58–0x7F: Reserved
	0x80–0xFF: PSC2 FIFOC registers
...	**...**
0xFF41_1800 (PSC8_BASE)	0x00–0x54: PSC8 registers
	0x58–0x7F: Reserved
	0x80–0xFF: PSC8 FIFOC registers
0xFF41_1900 (PSC9_BASE)	0x00–0x54: PSC9 registers
	0x58–0x7F: Reserved
	0x80–0xFF: PSC9 FIFOC registers
0xFF41_1A00–0xFF41_1EFF	Reserved
0xFF41_1F00 (FIFOC_BASE)	FIFOC registers

Table 26-2. PSC memory map

Offset from PSC_BASE[1] PSC0: 0xFF41_1000 PSC1: 0xFF41_1100 PSC2: 0xFF41_1200 PSC3: 0xFF41_1300 PSC4: 0xFF41_1400 PSC5: 0xFF41_1500 PSC6: 0xFF41_1600 PSC7: 0xFF41_1700 PSC8: 0xFF41_1800 PSC9: 0xFF41_1900	Register	Access	Reset Value	Section/Page
0x00–0x54	PSC registers. See Chapter 25, “Programmable Serial Controller (PSC).”			
0x58–0x7F	Reserved			
0x80	Command register for PSC*n* TX slice (PSC*n*_TX_CMD)	R/W	0x0000_0000	26.2.1.1/26-720
0x84	Alarm level for PSC*n* TX slice (PSC*n*_TX_ALARM)	R/W	0x0000_0000	26.2.1.2/26-721
0x88	Status register for PSC*n* TX slice (PSC*n*_TX_SR)	R	0x0000_0080	26.2.1.3/26-721
0x8C	Interrupt status register for PSC*n* TX slice (PSC*n*_TX_ISR)	R	0x0000_0000	26.2.1.4/26-722
0x90	Interrupt mask register for PSC*n* TX slice (PSC*n*_TX_IMR)	R/W	0x0000_0000	26.2.1.5/26-724
0x94	FIFO count for PSC*n* TX slice (PSC*n*_TX_COUNT)	R	0x0000_0000	26.2.1.6/26-725
0x98	FIFO pointer for PSC*n* TX slice (PSC*n*_TX_POINTER)	R	0x0000_0000	26.2.1.7/26-725
0x9C	FIFO size register for PSC*n* TX slice (PSC*n*_TX_SIZE)	R/W	0x0000_0000	26.2.1.8/26-726
0xA0–0xBB	Reserved			
0xBC	FIFO data register for PSC*n* TX slice (PSC*n*_TX_DATA)	R/W	0x0000_0000	26.2.1.9/26-726
0xC0	Command register for PSC*n* RX slice (PSC*n*_RX_CMD)	R/W	0x0000_0000	26.2.1.1/26-720
0xC4	Alarm level for PSC*n* RX slice (PSC*n*_RX_ALARM)	R/W	0x0000_0000	26.2.1.2/26-721
0xC8	Status register for PSC*n* RX slice (PSC*n*_RX_STAT)	R	0x0000_0080	26.2.1.3/26-721
0xCC	Interrupt status register for PSC*n* RX slice (PSC*n*_RX_ISR)	R	0x0000_0000	26.2.1.4/26-722
0xD0	Interrupt mask register for PSC*n* RX slice (PSC*n*_RX_IMR)	R/W	0x0000_0000	26.2.1.5/26-724

Table 26-2. PSC memory map (Continued)

Offset from PSC_BASE[1] PSC0: 0xFF41_1000 PSC1: 0xFF41_1100 PSC2: 0xFF41_1200 PSC3: 0xFF41_1300 PSC4: 0xFF41_1400 PSC5: 0xFF41_1500 PSC6: 0xFF41_1600 PSC7: 0xFF41_1700 PSC8: 0xFF41_1800 PSC9: 0xFF41_1900	Register	Access	Reset Value	Section/Page
0xD4	FIFO count for PSC*n* RX slice (PSC*n*_RX_COUNT)	R	0x0000_0000	26.2.1.6/26-725
0xD8	FIFO pointer for PSC*n* RX slice (PSC*n*_RX_POINTER)	R	0x0000_0000	26.2.1.7/26-725
0xDC	FIFO size register for PSC*n* RX slice (PSC*n*_RX_SIZE)	R/W	0x0000_0000	26.2.1.8/26-726
0xE0–0xEB	Reserved			
0xFC	FIFO data register for PSC*n* RX slice (PSC*n*_RX_DATA)	R/W	0x0000_0000	26.2.1.9/26-726

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, "System Configuration and Memory Map (XLBMEN + Mem Map)."

Table 26-3. FIFOC memory map

Offset from FIFOC_BASE (0xFF41_1F00)[1]	Register	Access	Reset Value	Section/Page
0x00	FIFOC Command Register (FIFOC_CMD)	R/W	0x0000_0000	26.2.1.10/26-727
0x04	FIFOC interrupt register (FIFOC_INT)	R	0x0000_0000	26.2.1.11/26-727
0x08	FIFOC DMA request register (FIFOC_DMA)	R	0x0000_0000	26.2.1.12/26-728
0x0C	Reserved			
0x10	FIFOC debug register (FIFOC_DEBUG)	R/W	0x0000_0000	26.2.1.13/26-728
0x14–0xFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, "System Configuration and Memory Map (XLBMEN + Mem Map)."

26.2.1 Register Descriptions

26.2.1.1 Command Register (CMD)

Address: Base + 0x80 (PSC*n*_TX_CMD)
Base + 0xC0 (PSC*n*_RX_CMD)

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0		0	
W								EOF	RESET SLICE					DMA_EN		SLICE_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-2. Command Register (CMD)

Table 26-4. CMD field descriptions

Field	Description
EOF	End of Frame. This bit is set by writing a 1 to this bit field, writing zero will be ignored. The next write access to the TX data register (PSCn_TX_DATA) will clear this bit. 0 The next data word is a normal data word. 1 Defines the next data word written to the data register as the last word of this frame.
RESET SLICE	Reset the FIFO slice. 0 Unused 1 Clears the underrun, overrun, and memory access error bits. Clears all data in the FIFO slice and reset the internal pointer.
DMA_EN	Enable the request to the DMA engine 0 Request to the DMA engine is disabled. 1 Request to the DMA engine is enabled. The FIFO controller generates a request for this slice to DMA engine if this slice is enabled and the current data pointer reaches the alarm level.
SLICE_EN	Enable this Slice of the FIFO 0 FIFO slice is disabled. All interrupt and request lines are cleared. Access to the internal register is possible, but the PSC cannot read or write data to the FIFO. 1 FIFO slice is enabled. The FIFO controller provides the data for the transmitter and stores the data from the receiver. If the size of the FIFO slice is zero, it's not possible to enable the FIFO.

26.2.1.2 Alarm Level (ALARM)

Address: Base + 0x84 (PSC*n*_TX_ALARM)
Base + 0xC4 (PSC*n*_RX_ALARM)

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ALARM LEVEL											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-3. Alarm Level Register (ALARM)

Table 26-5. ALARM field descriptions

Field	Description
ALARM LEVEL	The alarm level defines the number of data bytes in the FIFO when the alarm status appears. Also, the interrupt lines to the core or the request lines to the DMA are asserted if these lines are enabled. For the TX FIFO area, the alarm status appears if the number of data in the TX FIFO is below this alarm level. For the RX FIFO area, the alarm status appears if the number of data in the RX FIFO is above this alarm level. The alarm level and the request lines deassert if the number of data cross this level again.

26.2.1.3 Status Register (SR)

The Status register (SR) shows the internal status of the FIFO slice. If an event occurs, the corresponding bit in the SR and in the ISR register is set. If the event disappears, the bit in the SR is also cleared but the bit in the ISR register is active until it is cleared by software.

Address: Base + 0x88 (PSC*n*_TX_SR)
Base + 0xC8 (PSC*n*_RX_SR)

Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DEBUG MODE	0	0	0	0	0	0	0	SIZE ZERO	MEM ERROR	DATA READY	OR ERR	UR ERR	ALARM	FULL	EMPTY
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 26-4. Status Register (SR)

Table 26-6. SR field descriptions

Field	Description
DEBUG MODE	Debug Mode 0 The Debug mode is disabled. 1 The Debug mode is enabled.
SIZE ZERO	Size Zero 0 The programmed number of available memory words for this slice inside the Size register is not zero. 1 The programmed number of available memory words for this slice inside the Size register is zero.
MEM ERROR	Memory Access Error 0 No Error occurred. 1 The access to the data register generates an access error.
DATA READY	Data Ready 0 FIFO empty. 1 FIFO contains one or more data words.
ORERR	Overrun Error 0 No overrun error. 1 Overrun error occurred, write access to a full FIFO. **Note:** The PSC will not write to a full FIFO to avoid data lost. Therefore a PSC access cannot generate a ORERR situation. To see an overrun error event during a PSC transmission, software must check the ORERR bit in the PSC ISR register.
UR ERR	Underrun Error 0 No underrun error. 1 Underrun error occurred, read access from an empty FIFO. **Note:** The PSC will not read from an empty FIFO to avoid data lost. Therefore a PSC access cannot generate a URERR situation. To see an underrun error event during a PSC transmission, software must check the URERR bit in the PSC ISR register.
ALARM	FIFO Alarm 0 The number of data in the FIFO has not reached the alarm level. 1 The number of data in the FIFO has reached the alarm level.
FULL	FIFO Full 0 The FIFO is not full. 1 The FIFO is full.
EMPTY	FIFO Empty 0 The FIFO contains data. 1 The FIFO is empty.

26.2.1.4 Interrupt Status Register (ISR)

Writing one to the interrupt bit clears the interrupt bit.

Address: Base + 0x8C (PSC*n*_TX_ISR)
Base + 0xCC (PSC*n*_RX_ISR)

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	MEM ERROR	DATA READY	OR ERR	UR ERR	ALARM	FULL	EMPTY
W										w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-5. Interrupt Status Register (ISR)

Table 26-7. ISR field descriptions

Field	Description
MEM ERROR	Memory Access Error. This bit is identical to the MEM error bit in the SR register. If the corresponding bit in the IMR register also set, an interrupt is generated. Writing 1 to this bit clears the interrupt.
DATA READY	Data Ready. This is identical to the DATA READY bit in the SR register. If the corresponding bit in the IMR register is also set, an interrupt is generated. Writing 1 to this bit clears the interrupt.
ORERR	Overrun Error. This bit is identical to the ORERR bit in the SR register. If the corresponding bit in the IMR register also set, an interrupt is generated. Writing 1 to this bit clears the interrupt.
URERR	Underrun Error. This bit is identical to the ORERR bit in the SR register. If the corresponding bit in the IMR register is also set, an interrupt is generated. Writing 1 to this bit clears the interrupt.
ALARM	FIFO Alarm. This bit is identical to the ALARM bit in the SR register. If the corresponding bit in the IMR register is also set, an interrupt is generated. Writing 1 to this bit clears the interrupt.
FULL	FIFO Full. This bit is identical to the FULL bit in the SR register. If the corresponding bit in the IMR register is also set, an interrupt is generated. Writing 1 to this bit clears the interrupt.
EMPTY	FIFO Empty. This bit is identical to the EMPTY bit in the SR register. If the corresponding bit in the IMR register is also set, an interrupt is generated. Writing 1 to this bit clears the interrupt.

26.2.1.5 Interrupt Mask Register (IMR)

Address: Base + 0x90 (PSC*n*_TX_IMR)
Base + 0xD0 (PSC*n*_RX_IMR)

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	MEM ERROR	DATA READY	OR ERR	UR ERR	ALARM	FULL	EMPTY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-6. Interrupt Mask Register (IMR)

Table 26-8. IMR field descriptions

Field	Description
MEM ERROR	Memory Access Error 0 The memory access error status has no effect on the interrupt line. 1 Enable the interrupt for memory access error.
DATA READY	Data Ready 0 The DATA READY status has no effect on the interrupt line. 1 Enable the interrupt for DATA READY status.
ORERR	Overrun Error 0 The overrun error status has no effect on the interrupt line. 1 Enable the interrupt for overrun error.
URERR	Underrun Error 0 The underrun error status has no effect on the interrupt line. 1 Enable the interrupt for underrun error.
ALARM	FIFO Alarm 0 The FIFO alarm status has no effect on the interrupt line. 1 Enable the interrupt for FIFO alarm.
FULL	FIFO Full 0 The FIFO full status has no effect on the interrupt line. 1 Enable the interrupt for FIFO full.
EMPTY	FIFO Empty 0 The FIFO empty status has no effect on the interrupt line. 1 Enable the interrupt for FIFO empty.

26.2.1.6 Count Register (COUNT)

Address: Base + 0x94 (PSC*n*_TX_COUNT)
Base + 0xD4 (PSC*n*_RX_COUNT)

Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	COUNT												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-7. Count Register (COUNT)

Table 26-9. COUNT field descriptions

Field	Description
COUNT	Count. This read-only register shows the number of bytes in the FIFO.

26.2.1.7 Pointer Register (PTR)

Address: Base + 0x98 (PSC*n*_TX_PTR)
Base + 0xD8 (PSC*n*_RX_PTR)

Access: User read-only / Debug write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	WRITE POINTER												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	READ POINTER												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-8. Pointer Register (PTR)

Table 26-10. PTR field descriptions

Field	Description
WRITE POINTER	Write Pointer. This read-only register shows the current write position in the FIFO memory without the offset for the slice number. This register is writeable during debug mode.
READ POINTER	Read Pointer. This read-only register shows the current read position in the FIFO memory without the offset for the slice number. This register is writeable during debug mode.

26.2.1.8 FIFO Size Register (SIZE)

Address: Base + 0x9C (PSC*n*_TX_SIZE)
Base + 0xDC (PSC*n*_RX_SIZE)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	FIFO START ADDRESS									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	FIFO SIZE										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-9. FIFO Size Register (SIZE)

Table 26-11. SIZE field descriptions

Field	Description
FIFO START ADDRESS	Defines the word start address for this FIFO slice. The software must take care not to allow overlap in the memory areas to appear.
FIFO SIZE	Defines the size (number of words) of this FIFO slice.

26.2.1.9 Data Register (DATA)

Address: Base + 0xBC (PSC*n*_TX_SIZE)
Base + 0xFC (PSC*n*_RX_SIZE)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-10. Data Register (DATA)

Table 26-12. DATA field descriptions

Field	Description
DATA	Data Register. Write access to this register writes the data to the FIFO. Read access from this register reads the data from the FIFO.

26.2.1.10 FIFOC Command Register (FIFOC_CMD)

Address: FIFOC Base + 0x00 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CLOCK_GATE_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-11. FIFOC Command Register (FIFOC_CMD)

Table 26-13. FIFOC_CMD field descriptions

Field	Description
CLOCK_GATE_EN	Dynamic Clock Gating Enabled 0 Dynamic clock gating is disabled 1 Dynamic clock gating is enabled, the FIFO gates off the internal clock if no access is available. This behavior increases the number of IP bus wait cycles by one.

26.2.1.11 FIFOC Interrupt Register (FIFOC_INT)

Address: FIFOC Base + 0x04 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R							FIFOC RX INTERRUPT[PSC9:PSC0]									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R							FIFOC TX INTERRUPT[PSC9:PSC0]									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-12. FIFOC Interrupt Register (FIFOC_INT)

Table 26-14. FIFOC_INT field descriptions

Field	Description
FIFOC RX INTERRUPT	FIFOC RX Interrupt. There is one bit per PSC FIFO showing all PSCs with currently pending interrupts.
FIFOC TX INTERRUPT	FIFOC TX Interrupt. There is one bit per PSC FIFO showing all PSCs with currently pending interrupts.

26.2.1.12 FIFOC DMA Request Register (FIFOC_DMA)

Address: FIFOC Base + 0x08 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R							FIFOC RX DMA REQUEST[PSC9:PSC0]									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R							FIFOC TX DMA REQUEST[PSC9:PSC0]									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-13. FIFOC DMA Request Register (FIFOC_DMA)

Table 26-15. FIFOC_DMA field descriptions

Field	Description
FIFOC RX DMA REQUEST	FIFOC RX DMA REQUEST. There is one bit per PSC FIFO showing all PSCs with currently pending requests.
FIFOC TX DMA REQUEST	FIFOC TX DMA REQUEST. There is one bit per PSC FIFO showing all PSCs with currently pending requests.

26.2.1.13 FIFOC Debug Register (FIFOC_DEBUG)

Address: FIFOC Base + 0x10 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LOCK	DE BUG
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-14. FIFOC Debug Register (FIFOC_DEBUG)

Table 26-16. FIFOC_DEBUG field descriptions

Field	Description
DEBUG	Debug Mode 0 Normal Operation Mode 1 Debug mode. The Interrupt status register and the pointer register are writeable. The debug bit is only writeable with a 32-bit access to this register during bit[31:16] containing 0x8442.
LOCK	State Machine Lock 0 Normal operation mode 1 Lock the internal machine control state machine. All PSC requests are ignored. Only access from the IP Bus interface is possible.The lock bit is only writeable with a 32-bit access to this register during bit[31:16] containing 0x8442.

26.3 Functional Description

During the functional mode, the FIFOC controller provides the data for all PSC transmitters and stores the data from all PSC receiver. If the FIFO gets a request from the PSC, the required data from the TX FIFO memory is written to the PSC transmit register or the received data is read from the RX register and is stored in the RX FIFO area. If the number of data inside the FIFO register reaches the programmed alarm level, the enabled request lines are asserted to inform the system that the RX data is available or new TX data is required.

This page is intentionally left blank.

Chapter 27
Real Time Clock (RTC)

27.1 Introduction

The real time clock module (RTC) provides a time of day function as well as a calendar function. It also provides a method to bring the MPC5125 from a power-down condition to a fully operational condition. The RTC provides a time base as long as V_{BAT} power is applied. Parts of the RTC module are powered independently of the rest of the MPC5125 device. The RTC has its own internal 32.768 kHz oscillator which is totally independent of the other MPC5125 clock domains. As long as V_{BAT} is powered, the RTC can be used to cause the MPC5125 to exit any of the power-down modes, which include Doze, Nap, Sleep, Deep Sleep, and Hibernation mode. The wakeup function can be initiated from the internal timer inside the RTC module or from any of six external wakeup pins. An external pin, $\overline{\text{HIB\_MODE}}$, is associated with the RTC and can be used to control the power supplies for the MPC5125. The RTC controls the $\overline{\text{HIB\_MODE}}$ pin in response to any RTC wakeup event, such as a transition on one of the six external wakeup sources or a timeout event in the RTC module.

A block diagram of the RTC is shown in Figure 27-1.

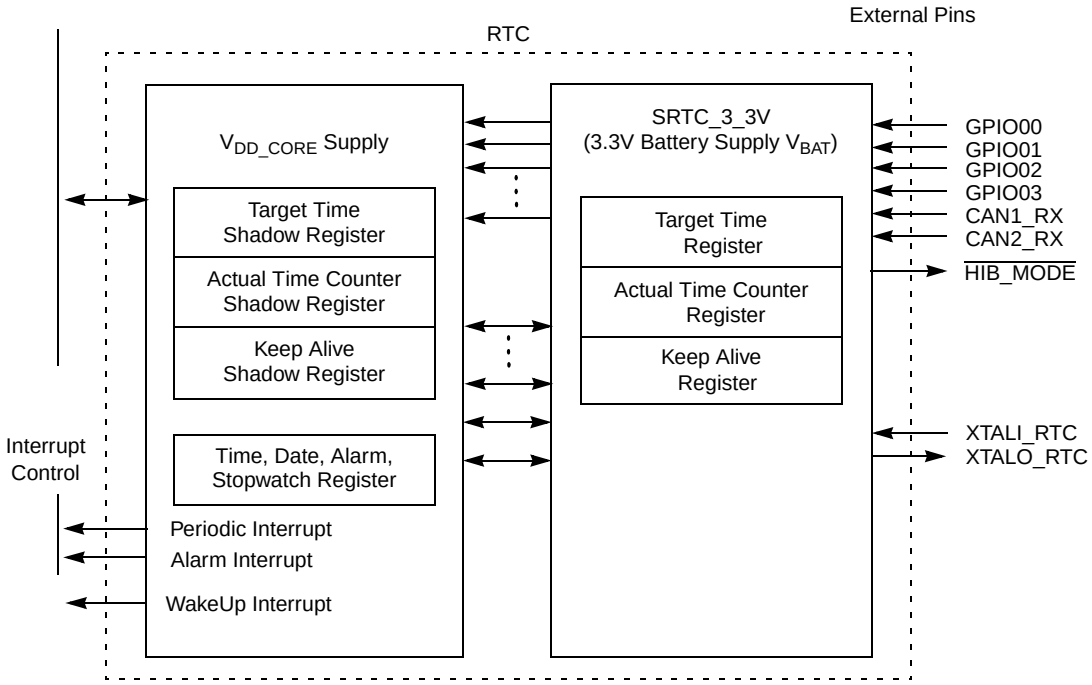


Figure 27-1. RTC Block Diagram

27.1.1 Features

The RTC module provides the following features:

- Secure real time clock (SRTC_3_3V)
  - Secure (not programmable) low-power timebase (actual time count register) that runs continuously as long as V_{BAT} is powered and the crystal is connected to RTC OSC
  - $\overline{\text{HIB\_MODE}}$ (Hibernate) pin used to enable/disable external power regulator
  - Programmable timeout and six external wakeup sources, for exiting $\overline{\text{HIB\_MODE}}$
    - Programmable wakeup source edge detect
    - Programmable wakeup source inhibitor after the first wakeup source has been detected
  - Wakeup source bits and four additional bits in RTC keep alive register can be used as an 8-bit breadcrumb register when the MPC5125 is powered off
  - Loss of power bit, TAMP
- Normal RTC functions under software control
  - Minute countdown timer—provides 256-minute capability, slightly over 4 hours
  - Programmable alarm—operates on time of day only, not related to calendar
  - Periodic interrupts for:
    - 1 second
    - 1 minute
    - 1 day—operates only at midnight rollover
  - Calendar features:
    - Weekday
    - Date
    - Year
- Crystal support (32.768 kHz only)

The RTC is divided into two major sections. One section contains the target time shadow register, actual time counter shadow register, keep alive shadow register and the time, date, alarm, and stopwatch registers. This section is powered by the V_{DD_CORE} power supply. These registers retain their contents when powered by the V_{DD_CORE} power supply. If the V_{DD_CORE} power supply is turned off, these registers lose their contents.

To provide for the restoration of the date and time after all power is lost except for V_{BAT}, the system software needs to record a value of the actual time count register that corresponds with the contents of the RTC current time register and current date register. These values must be stored in non-volatile memory. Then, after power is restored, software must be used to deduce the current date and time based on the current count of the actual time count register.

The time, date and stopwatch registers are updated by the 1-second clock that is derived from the RTC 32.768 kHz oscillator. When the MPC5125 enters the deep sleep mode, the contents of these registers are maintained and updated as long as the V_{DD_CORE} supply is powered. That is, these registers are updated by the 1-second clock derived from the RTC 32.768 kHz oscillator; however, when in the deep sleep mode,

the real time clock cannot generate a wakeup interrupt based on an alarm time to cause the MPC5125 to exit the deep sleep mode. The deep sleep mode can only be exited in response to an external RTC wakeup pin or when the actual time count register is greater than or equal to the target time register.

The second section of the RTC contains the target time register, actual time counter register, and the keep alive register. This section of the RTC is powered by the V_{BAT} supply. As long as power is applied to the V_{BAT} pin, the actual time counter register continues to increment at a 1 Hz rate.

When the CPU reads the value of the actual time count register, target time register or the keep alive register, the value in the respective shadow register is actually fetched. The contents of the actual time count register, target time register or the keep alive register are transferred to their respective shadow registers on each cycle of the 1-second clock which increments the actual time count register.

For predictable operation, consecutive writes to the actual time count shadow register, the target time shadow register or the keep alive shadow register must be separated by at least three clock periods of the 32.768 kHz RTC clock.

27.2 External Signal Descriptions

Table 27-1. External Pins Signal Properties

Name	Function	I/O	Reset
RTC_XTALI	32 kHz crystal input	I	—
RTC_XTALO	32 kHz crystal output	O	—
$\overline{\text{HIB\_MODE}}$	Power regulator disable (See Table 27-14 and Table 27-16)	O	1
GPIO00	General purpose input	I	—
GPIO01	General purpose input	I	—
GPIO02	General purpose input	I	—
GPIO03	General purpose input	I	—
CAN1_RX	CAN Receive input	I	—
CAN2_RX	CAN Receive input	I	—

27.3 Memory Map and Register Definition

27.3.1 Memory Map

Table 27-2 shows the memory map for the MPC5125 RTC.

Table 27-2. RTC memory map

Offset from RTC_BASE (0xFF40_0A00)[1]	Register	Access	Reset Value	Section/Page
0x00	RTC_TSR—RTC Time Set Register	R/W	0x0000_0000	27.3.2.1/27-734
0x04	RTC_DSR—RTC Date Set Register	R/W	0x0000_0000	27.3.2.2/27-736

Table 27-2. RTC memory map (Continued)

Offset from RTC_BASE (0xFF40_0A00)[1]	Register	Access	Reset Value	Section/Page
0x08	RTC_NY_STP—RTC New Year and Stopwatch Register	R/W	0x0000_0000	27.3.2.3/27-737
0x0C	RTC_ALM_IE—RTC Alarm and Interrupt Enable Register	R/W	0x0000_0008	27.3.2.4/27-738
0x10	RTC_CTR—RTC Current Time Register	R	0x0000_0000	27.3.2.5/27-739
0x14	RTC_CDR—RTC Current Date Register	R	0x0000_0000	27.3.2.6/27-739
0x18	RTC_ALM_STP_INT—RTC Alarm and Stopwatch Interrupt Register	R/W	0x0000_0000	27.3.2.7/27-740
0x1C	RTC_PI_BE—RTC Periodic Interrupt and Bus Error Register	R/W	0x0000_0000	27.3.2.8/27-741
0x20	RTC_TTR—RTC Target Time Register	R/W	0x0000_0000	27.3.2.9/27-741
0x24	RTC_ATC—RTC Actual Time Counter Register	R	0xFFFF_FFFE	27.3.2.10/27-742
0x28	RTC_KAR—RTC Keep Alive Register	R/W	0x0000_0087	27.3.2.11/27-743
0x2C–0xFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

NOTE

All registers other than the RTC Target Time register (RTC_TTR), the RTC Actual Time Counter Register (RTC_ATC), and the RTC Keep Alive Register (RTC_KAR) registers lose content when V_{DD_CORE} power is no longer supplied. These three registers are powered by V_{BAT}.

27.3.2 Register Descriptions

27.3.2.1 RTC Time Set Register (RTC_TSR)

The RTC Time Set Register (RTC_TSR) is used to program the RTC current time register (see Section 27.3.2.5, “RTC Current Time Register (RTC_CTR)”). This register does not reflect the running time value.

Address: Base + 0x00 Access: User read/write

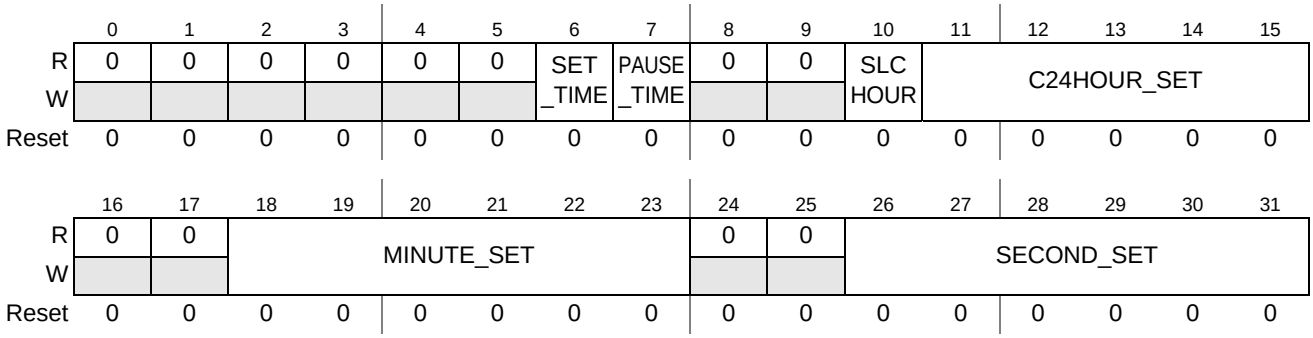


Figure 27-2. RTC Time Set Register (RTC_TSR)

Table 27-3. RTC_TSR field descriptions

Field	Description
SET_TIME	A software sequence using the pause time bit in conjunction with the set_time bit must be followed to load new values into the RTC current time register (see Section 27.3.2.5, "RTC Current Time Register (RTC_CTR)"). The SET_TIME bit cannot be set to 1 unless the PAUSE_TIME bit is also set to 1. The proper software sequence is: 1. Write register with C24HOUR_SET, MINUTE_SET, and SECOND_SET fields set to the desired values and with PAUSE_TIME = 1 and SET_TIME = 0. 2. Write register with C24HOUR_SET, MINUTE_SET, and SECOND_SET fields set to the desired values and with PAUSE_TIME = 1 and SET_TIME = 1. 3. Write register with C24HOUR_SET, MINUTE_SET, and SECOND_SET fields set to the desired values and with PAUSE_TIME = 1 and SET_TIME = 0. 4. Write register with C24HOUR_SET, MINUTE_SET, and SECOND_SET fields set to the desired values and with PAUSE_TIME = 0 and SET_TIME = 0. At completion of step 4, the RTC current time register is updated with the new time. The C24HOUR_SET, MINUTE_SET, and the SECOND_SET fields should remain consistent throughout the four steps (i.e., at the desired new time values). The hour, minute and second fields of the RTC current time register are all set at the completion of step 4 of the previous sequence. It is important to use LOAD and STORE operations to access this register. Do **not** use read-modify-write instructions such as AND or OR to modify this register.
PAUSE_TIME	Used with set_time above to perform time update. Must be 0 for normal operation.
SLCHOUR	This bit determines the hour output format. 0 24-hour format. 1 12-hour format with AM/PM. This bit does *not* affect time set procedure, it only affects how the hour status field is presented. The new hour format is updated immediately in the current time register.
C24HOUR_SET	Hour in 24-hour format written in RTC current time register after successful state machine transition by set_time and pause_time bits. This field is always written with 24-Hour format, it is *not* affected by SLCHOUR bit above.
MINUTE_SET	Minute written in RTC current time register after successful state machine transition by SET_TIME and PAUSE_TIME bits.
SECOND_SET	Second written in RTC current time register after successful state machine transition by SET_TIME and PAUSE_TIME bits.

27.3.2.2 RTC Date Set Register (RTC_DSR)

The RTC Date Set Register (RTC_DSR) is used to program the RTC current date register (see Section 27.3.2.6, “RTC Current Date Register (RTC_CDR)”). The RTC date set register does not reflect the running date value. Notice that the value in year_set field of the RTC new year and stopwatch register (see Section 27.3.2.3, “RTC New Year and Stopwatch Register (RTC_NY_STP) Register”) is also loaded into the current date register after the proper software sequence using the set_date and pause_date bits.

Address: Base + 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	SET_DATE	PAUSE_DATE	0	0	0	MONTH_SET				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	WEEKDAY_SET						0	0	DATE_SET					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-3. RTC Date Set Register (RTC_DSR)

Table 27-4. RTC_DSR field descriptions

Field	Description
SET_DATE	A software sequence using the pause_date bit in conjunction with the set_date bit must be followed to load new values into the RTC current time register (Section 27.3.2.5, “RTC Current Time Register (RTC_CTR)”). The SET_DATE bit cannot be set to 1 unless the PAUSE_DATE bit is also set to 1. The proper software sequence is: • Step 1. Write register with MONTH_SET, WEEKDAY_SET, DATE_SET, and YEAR_SET fields set to the desired values and with PAUSE_DATE = 1 and SET_DATE = 0. • Step 2. Write register with MONTH_SET, WEEKDAY_SET, DATE_SET, and YEAR_SET fields set to the desired values and with PAUSE_DATE = 1 and SET_DATE = 1. • Step 3. Write register with MONTH_SET, WEEKDAY_SET, DATE_SET, and YEAR_SET fields set to the desired values and with PAUSE_DATE = 1 and SET_DATE = 0. • Step 4. Write register with MONTH_SET, WEEKDAY_SET, DATE_SET, and YEAR_SET fields set to the desired values and with PAUSE_DATE = 0 and SET_DATE = 0. • At completion of Step 4, RTC Current date register is updated with the new time. The MONTH_SET, WEEKDAY_SET, and the DATE_SET fields should remain consistent throughout the four steps (i.e., at the desired new time values). The YEAR_SET field is located in the RTC new year and stopwatch register. The month, weekday, date and year fields are all set at the completion of step 4 of the previous sequence. It is important to use LOAD and STORE operations to access this register. Do not use read-modify-write instructions such as AND or OR to modify this register.
PAUSE_DATE	Used with set_time above to perform time update. Must be zero for normal operation.
MONTH_SET	New month written in RTC current date register after successful state machine transition by set_date and pause_date bits. Only the lower 4 bits of this field are used.
WEEKDAY_SET	New weekday written in RTC current date register after state machine transition by set_date and pause_date bits. 1 = Monday; 7 = Sunday. Only the lower 3 bits of this field are used. If 0 is written (not recommended), the weekday is set to 0 until the end of the day. The weekday is then incremented to 1.

Table 27-4. RTC_DSR field descriptions (Continued)

Field	Description
DATE_SET	New date (1-31) is written in RTC current date register after state machine transition by SET_DATE and PAUSE_DATE bits. Only the lower 5 bits of this field are used. A date of 1 indicates the first day of the month, 2 indicates the second day of the month, 31 indicates the last day of the month of January. If 0 is written (not recommend), the date is set to 0 until the end of the day. The date is then incremented to 1. YEAR_SET in the following register is also part of the date set function.

27.3.2.3 RTC New Year and Stopwatch Register (RTC_NY_STP) Register

Address: Base + 0x08 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	WRITE_SW	SW_SET							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	YEAR_SET											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-4. RTC New Year and Stopwatch (RTC_NY_STP) Register

Table 27-5. RTC_NY_STP field descriptions

Field	Description
WRITE_SW	Typical stopwatch operation is to write an initial value into the 8-bit wide SW_SET field and assert the WRITE_SW bit. The WRITE_SW bit is immediately auto cleared, but it triggers the stopwatch minute countdown to begin.
SW_SET	Number of minutes to be written into stopwatch. Max is 255, a little over 4 hours.
YEAR_SET	New year value to be written to the RTC current date register after successful state machine transition by SET_DATE and PAUSE_DATE bits. This is part of date set function in the previous register.

27.3.2.4 RTC Alarm and Interrupt Enable (RTC_ALM_IE) Register

Address: Base + 0x0C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	ALM_ENABLE	0	0	0	ALM_24H_SET				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	ALM_MIN_SET						0	0	0	0	MPE	INTEN_DAY	INTEN_MIN	INTEN_SEC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 27-5. RTC Alarm and Interrupt Enable (RTC_ALM_IE) Register

Table 27-6. RTC_ALM_IE field descriptions

Field	Description
ALM_ENABLE	Alarm enable bit for once-a-day alarm. 1 Alarm status/interrupt operation is enabled. 0 Alarm setting is not compared to time of day.
ALM_24H_SET	Hour setting (in 24 hour format) to be compared to time of day for the purpose of generating alarm status/interrupt. Can be written at any time.
ALM_MIN_SET	Minute setting to be compared to time of day for the purpose of generating alarm status/interrupt. Can be written at any time.
$\overline{\text{MPE}}$	Master periodic enable – Must be written low after reset to allow periodic interrupts. 0 Allow periodic interrupts. 1 Do not allow periodic interrupts.
INTEN_DAY	Enable bit of periodic interrupts at midnight rollover.
INTEN_MIN	Enable bit of periodic interrupts at minute rollover.
INTEN_SEC	Enable bit of periodic interrupts at second rollover.

NOTE

The Alarm function generates an interrupt based on a comparison to the hour and minute fields of the RTC current time register, shown in Figure 27-6. Thus, if the alarm function is enabled and left enabled, an alarm is generated each day at a particular time. It is important not to confuse the alarm interrupt with the midnight rollover interrupt. Either one can cause an interrupt once each 24 hours.

When in deep sleep, the alarm function cannot be used to exit the deep sleep mode.

27.3.2.5 RTC Current Time Register (RTC_CTR)

Address: Base + 0x10 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	HOUR				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	MINUTE						0	0	SECOND					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-6. RTC Current Time Register (RTC_CTR)

Table 27-7. RTC_CTR field descriptions

Field	Description
HOUR	Hour format can be either 24-hour or 12-hour with AM/PM. If 24-hour format is selected (SlctHour low in Hour[0]), the whole 5-bit hour field designates current time in 24-hour format. If 12-hour format is selected (SlctHour high in Hour[0]), the MSB of hour field indicates: Hour[0] =0: AM Hour[0] = 1: PM and Hour[1:4] designates current time in 12-hour format.
MINUTE	Shows minutes in current time.
SECOND	Shows seconds in current time.

27.3.2.6 RTC Current Date Register (RTC_CDR)

Address: Base + 0x14 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	MONTH				WEEKDAY			DATE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	YEAR											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-7. RTC Current Date Register (RTC_CDR)

Table 27-8. RTC_CDR field descriptions

Field	Description
MONTH	Shows current month. (1 = January; 12 = December)
WEEKDAY	Indicates day of week. (Monday = 1, Sunday = 7)
DATE	Shows current date. Calendar feature is implemented, therefore, day rollover at the end of month including February (and Leap Years) is automatic.
YEAR	Shows current year. Max is 4052.

27.3.2.7 RTC Alarm and Stopwatch Interrupt (RTC_ALM_STP_INT) Register

Address: Base + 0x18 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	INT_ALM	0	0	0	0	0	0	0	INT_SW
W								w1c								w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	ALM_STATUS	SW_MIN							
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-8. RTC Alarm and Stopwatch Interrupt (RTC_ALM_STP_INT) Register

Table 27-9. RTC_ALM_STP_INT field descriptions

Field	Description
INT_ALM	Status bit indicating that enabled once-a-day alarm has occurred (active high). Alarm interrupt has been activated. This bit and the Interrupt are cleared by writing 1 to this bit position. A stopwatch interrupt, if also active, must be cleared before the interrupt signal to the CPU is negated.
INT_SW	Status bit indicating that stopwatch expiration has occurred (active high). Stopwatch interrupt has been activated. This bit and the Interrupt are cleared by writing 1 to this bit position. An alarm interrupt, if also active, must be cleared before the interrupt signal to the CPU is negated.
ALM_STATUS	Status bit indicating that once-a-day alarm has occurred. Same as INT_ALM bit above except that clearing this bit does not clear the interrupt.
SW_MIN	Minutes remaining in stopwatch.

27.3.2.8 RTC Periodic Interrupt and Bus Error (RTC_PI_BE) Register

Address: Base + 0x1C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	BUS_ERROR_1	0	0	0	0	0	0	0	INT_DAY
W								w1c								w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	INT_MIN	0	0	0	0	0	0	0	INT_SEC
W								w1c								w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-9. RTC Periodic Interrupt and Bus Error (RTC_PI_BE) Register

Table 27-10. RTC_PI_BE field descriptions

Field	Description
BUS_ERROR_1	Internal status register—If high, indicates software has attempted a write access to a read-only register in this module. No actual register contents are corrupted and no interrupt is generated if this happens. Cleared by writing 1 to this bit position.
INT_DAY	Periodic interrupt at midnight. High indicates interrupt has occurred. Cleared by writing 1 to this bit position.
INT_MIN	Periodic interrupt at each minute rollover. High indicates interrupt has occurred. Cleared by writing 1 to this bit position.
INT_SEC	Periodic interrupt at each second rollover. High indicates interrupt has occurred. Cleared by writing 1 to this bit position.

27.3.2.9 RTC Target Time Register (RTC_TTR)

This register is used to program a time to hibernate, in seconds, based on value in the RTC current time register. When the RTC_ATC register is greater than or equal to the RTC time target register, the $\overline{\text{HIB\_MODE}}$ signal asserts to a logic 1.

Address: Base + 0x20 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TTR[0:15]															
Reset[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TTR[16:31]															
Reset[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-10. RTC Target Time Register (RTC_TTR)

1 This register is reset only when V_{BAT} is connected. After that, software must ensure it is updated correctly.

Table 27-11. RTC_TTR field descriptions

Field	Description
TTR	The number, in seconds, to generate the $\overline{\text{HIB\_MODE}}$ signal. When RTC_TTR = 0x0000_0000, entering Hibernation or Deep Sleep mode is not possible. If it is desired to use one of the wakeup sources to wakeup (Section 27.3.2.11, "RTC Keep Alive Register (RTC_KAR)"), then write RTC_TTR to 0xFFFF_FFFF. This allows to assert the $\overline{\text{HIB\_MODE}}$ signal and to enter Deep Sleep mode, when non of the external wake up sources (GPIO[00:03] or CAN[1:2]_RX) prevent this.

NOTE

Because of the uncertainty of the time when the time counter increments, it is recommended that the target time register be written with a value that is at least two counts greater than the current value of the RTC_ATC register.

27.3.2.10 RTC Actual Time Counter (RTC_ATC) Register

The RTC Actual Time Counter (RTC_ATC) register is used to read the elapsed time, in seconds, since V_{BAT} was last connected.

Address: Base + 0x24 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ATC[0:15]															
W																
Reset[1]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ATC[16:31]															
W																
Reset[1]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Figure 27-11. RTC Actual Time Counter (RTC_ATC) register

1 This register is reset only when V_{BAT} is connected. Software cannot modify this register.

Table 27-12. RTC_ATC field descriptions

Field	Description
ATC	The time, in seconds, since V_{BAT} was last connected.

NOTE

This register is set to 0xFFFF_FFFE when V_{BAT} is applied. If V_{BAT} is removed and later connected, the register is again set to 0xFFFF_FFFE.

The contents of this register cannot be modified by software. After V_{BAT} is connected, this register increments at a 1 Hz rate starting from 0xFFFF_FFFE.

27.3.2.11 RTC Keep Alive Register (RTC_KAR)

The RTC Keep Alive Register (RTC_KAR) contains a unique 9-bit breadcrumb register (BC_[0-8]), which can also function as a 5-bit wakeup source register (GPIO[01:03] and CAN[1:2]_RX) and a 4-bit breadcrumb register (BC_[5-8]). Each bit that functions as a wakeup source can also function as a breadcrumb bit. This register also contains a dedicated wakeup source register (GPIO00).

Breadcrumb bits are generally used to hold status words or information that is user defined. As long as V_{BAT} is powered, this register retains its contents. One purpose for the breadcrumb bits might be to hold user defined information about what process should be executed upon exiting the HIBERNATE mode.

NOTE

When writing to the RTC keep alive register, do not modify any of the WU_SRC_[1:5]_EN bits in the same instruction that modifies any of the WU_SRC_[1:5]_LVL bits. Use one instruction to modify the WU_SRC_[1:5]_LVL bits and a second instruction to modify the WU_SRC_[1:5]_EN bits.

Address: Base + 0x28 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	BC_7	WU_SRC_[1:5]_EN					0	0	BC_8	WU_SRC_[1:5]_LVL				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WU_SRC_0	WU_SRC_[1:5]					BC_5	BC_6	DIS_HIB_MODE	0	0	0	0	WU_SRC_MODE	TAMP	TTR_CMP
W	w1c													w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1

Figure 27-12. RTC Keep Alive Register (RTC_KAR)

Table 27-13. RTC_KAR field descriptions

Field	Description
BC_[8:5]	Breadcrumb bits. These bits are cleared when V_{BAT} is applied.
WU_SRC_1_EN	This bit determines the modes of the WU_SRC_1 bit. 0 WU_SRC_1 functions as a breadcrumb bit (BC_0). 1 WU_SRC_1 functions as a wakeup source (GPIO01).
WU_SRC_2_EN	This bit determines the modes of the WU_SRC_2 bit. 0 WU_SRC_2 functions as a breadcrumb bit (BC_1). 1 WU_SRC_2 functions as a wakeup source (GPIO02).
WU_SRC_3_EN	This bit determines the modes of the WU_SRC_3 bit. 0 WU_SRC_3 functions as a breadcrumb bit (BC_2). 1 WU_SRC_3 functions as a wakeup source (GPIO03).
WU_SRC_4_EN	This bit determines the modes of the WU_SRC_4 bit. 0 WU_SRC_4 functions as a breadcrumb bit (BC_3). 1 WU_SRC_4 functions as a wakeup source (CAN2_RX).
WU_SRC_5_EN	This bit determines the modes of the WU_SRC_5 bit. 0 WU_SRC_5 functions as a breadcrumb bit (BC_4). 1 WU_SRC_5 functions as a wakeup source (CAN1_RX).
WU_SRC_1_LVL	Program the level of the wakeup source, GPIO01, as active high or active low. 0 GPIO01 is active low. 1 GPIO01 is active high. **Note:** This must be configured before WU_SRC_1_EN is asserted to 1.
WU_SRC_2_LVL	Program the level of the wakeup source, GPIO02, as active high or active low. 0 GPIO02 is active low. 1 GPIO02 is active high. **Note:** This must be configured before WU_SRC_2_EN is asserted to 1.
WU_SRC_3_LVL	Program the level of the wakeup source, GPIO03, as active high or active low. 0 GPIO03 is active low. 1 GPIO03 is active high. **Note:** This must be configured before WU_SRC_1_EN is asserted to 1.
WU_SRC_4_LVL	Program the level of the wakeup source, CAN2_RX, as active high or active low. 0 CAN2_RX is active low. 1 CAN2_RX is active high. **Note:** This must be configured before WU_SRC_1_EN is asserted to 1.
WU_SRC_5_LVL	Program the level of the wakeup source, CAN1_RX, as active high or active low. 0 CAN1_RX is active low. 1 CAN1_RX is active high. **Note:** This must be configured before WU_SRC_1_EN is asserted to 1.
WU_SRC_0	Bit functions as a wakeup source sticky bit. When GPIO00 is asserted low then this bit is set to 1. Software can clear it by writing a 1 to this location when GPIO00 is negated high.
WU_SRC_1	WU_SRC_1_EN = 0: Breadcrumb bit (BC_0) function WU_SRC_1_EN = 1: Bit functions as a wakeup source sticky bit. When GPIO01 is asserted, according to WU_SRC_1_LVL bit, then this bit is set to 1. Software can clear it by writing a 1 to this location when GPIO01 is its inactive wake up state. **Note:** This bit CANNOT be written at the same time WU_SRC_1_EN is written.

Table 27-13. RTC_KAR field descriptions (Continued)

Field	Description
WU_SRC_2	WU_SRC_2_EN = 0: Breadcrumb bit (BC_1) function WU_SRC_2_EN = 1: Bit functions as a wakeup source sticky bit. When GPIO02 is asserted, according to WU_SRC_2_LVL bit, then this bit is set to 1. Software can clear it by writing a 1 to this location when GPIO02 is its inactive wake up state. **Note:** This bit CANNOT be written at the same time WU_SRC_2_EN is written.
WU_SRC_3	WU_SRC_3_EN = 0: Breadcrumb bit (BC_2) function WU_SRC_3_EN = 1: Bit functions as a wakeup source sticky bit. When GPIO03 is asserted, according to WU_SRC_3_LVL bit, then this bit is set to 1. Software can clear it by writing a 1 to this location when GPIO03 is its inactive wake up state. **Note:** This bit CANNOT be written at the same time WU_SRC_3_EN is written.
WU_SRC_4	WU_SRC_4_EN = 0: Breadcrumb bit (BC_3) function WU_SRC_4_EN = 1: Bit functions as a wakeup source sticky bit. When CAN2_RX is asserted, according to WU_SRC_4_LVL bit, then this bit is set to 1. Software can clear it by writing a 1 to this location when CAN2_RX is its inactive wake up state. **Note:** This bit CANNOT be written at the same time WU_SRC_4_EN is written.
WU_SRC_5	WU_SRC_5_EN = 0: Breadcrumb bit (BC_4) function WU_SRC_5_EN = 1: Bit functions as a wakeup source sticky bit. When CAN1_RX is asserted, according to WU_SRC_4_LVL bit, then this bit is set to 1. Software can clear it by writing a 1 to this location when CAN1_RX is its inactive wake up state. **Note:** This bit CANNOT be written at the same time WU_SRC_4_EN is written.
DIS_HIB_MODE	This bit determines whether to disable the $\overline{\text{HIB\_MODE}}$ output pin. 0 Enable the $\overline{\text{HIB\_MODE}}$ output pin. 1 Disable the $\overline{\text{HIB\_MODE}}$ output pin.The $\overline{\text{HIB\_MODE}}$ output pin is always negated high regardless the wake up sources (TTR, GPIO[00:03], CAN[1:2]_RX).
WU_SRC_MODE	Wake Up Source Mode bit 0 Allow all proceeding wake up signals to be registered. 1 Inhibit all proceeding wake up signals after the first WU_SRC status bit is registered. **Note:** In cases two or more wake up signals occur within a 32 kHz window, both signals are registered in their corresponding WU_SRC bits even when WU_SRC_MODE is asserted to 1.
TAMP	This bit is set when V_{BAT} is applied or oscillator stops running. Software can clear this bit by writing an one to this bit. Software can use this bit recognize loss of V_{BAT} power or stop of RTC OSC.
TTR_CMP	This bit is the value of the RTC_TTR register compared to the RTC_ATC register. TTR_CMP = RTC_TTR < RTC_ATC

NOTE

- This register has no influence on the Deep Sleep mode, but does have influence on the Hibernation mode.

- The GPIO00 wake up source can neither be masked nor configured to a different polarity level. To enter Hibernation mode, this pin needs to be in a logic one state.

27.4 Functional Description

27.4.1 Behavior at Power On

When power is first applied to the RTC, the target time register is initialized to 0x0000_0000 and the actual time count register is initialized to 0xFFFF_FFFE. When the target time register is set to 0x0000_0000, the $\overline{\text{HIB\_MODE}}$ pin cannot be asserted. The Keep alive register is initialized to 0x0000_0087. This effectively masks off any external wakeup sources. To activate any of the external wakeup sources associated with the RTC module after power is first applied to V_{BAT}, software must appropriately program the RTC keep alive register.

After power is applied to the MPC5125, system software must initialize all of the various registers of the RTC module to properly reflect the current date and time.

27.4.2 Behavior of Wakeup Sources

The RTC employs two basic ways to bring the MPC5125 from a powerdown condition to a fully operational condition. One method is for the value in the actual time count register to increment to a value greater than or equal to the value in the target time register. The second method is to assert one of the external RTC wakeup pins.

From the release of reset, the time target register is set to 0x0000_0000. Thus, the actual time count register is always greater than or equal to the time target register. Under this condition, the $\overline{\text{HIB\_MODE}}$ asserts to a logic 1 and can be used to turn on external power regulators powering the device. Recall that the actual time count register is set to 0xFFFF_FFFE at the release of reset and cannot be modified by software. When being clocked at a 1 Hz rate, this register requires 136 years before the actual time count rollover. When programming a time in the future to wakeup, the system software reads the actual time count register, adds an appropriate offset and then writes the resultant value to the time target register.

When the value in the actual time count register is less than the value in the target time register, the logic level on the $\overline{\text{HIB\_MODE}}$ pin is determined by the WU_SRC_[0:5] bits in the RTC keep alive register. The truth table for controlling the $\overline{\text{HIB\_MODE}}$ pin is shown in Table 27-14 and Table 27-16.

Five of the external RTC wakeup pins are controlled by its own enable bit and edge detect bit. If an external RTC wakeup pin is not enabled, then its corresponding WU_SRC_[0:5] bit is held at a logic 0. If an external RTC wakeup pin is enabled, then its corresponding WU_SRC_[0:5] bit is set or cleared according to the truth table presented in Table 27-16.

Table 27-14. Truth Table for the Functionality of the $\overline{\text{HIB\_MODE}}$

Condition	WU_SRC_[0:5]	$\overline{\text{HIB\_MODE}}$
If RTC_ATC > RTC_TTR	X	1
If RTC_ATC = RTC_TTR	X	1

Table 27-14. Truth Table for the Functionality of the $\overline{\text{HIB\_MODE}}$ (Continued)

Condition	WU_SRC_[0:5]	$\overline{\text{HIB\_MODE}}$
If RTC_ATC < RTC_TTR	0	0
If RTC_ATC < RTC_TTR	1	1

Table 27-15. Truth Table for the Functionality of the RTC Wakeup Sources

WU_SRC_[1:5]_EN	WU_SRC_[1:5]_LVL	GPIO[01:03], CAN[1:2]_RX	WU_SRC_[1:5]
0	X	X	Breadcrumb function
1	0	$1 \rightarrow 0$	1
1	0	$0 \rightarrow 1$	Can be cleared by software
1	1	$0 \rightarrow 1$	1
1	1	$1 \rightarrow 0$	Can be cleared by software

Table 27-16. Truth Table for the Functionality of GPIO00 Wakeup Sources

GPIO00	WU_SRC_0
$1 \rightarrow 0$	1
$0 \rightarrow 1$	Can be cleared by software

In case two or more SET_WU_SRC signals occur simultaneously in a 32 kHz window, both signals are registered in their corresponding WU_SRC bits even when WU_SRC_MODE is asserted to 1.

27.4.3 Behavior During Power Off (Hibernation Mode)

The behavior of the RTC is illustrated in Figure 27-13 which shows power being applied to the device, removing power by asserting the $\overline{\text{HIB\_MODE}}$ pin and then returning to a power-on condition. Power is applied to the V_{BAT} pin. This causes the internal RTC_POR signal to the RTC to be released. RTC_POR, shown in Figure 27-13, puts the RTC in its active state when the internal RTC_POR signal is at a logic 0. The Actual Time Count (RTC_ATC) register is reset to 0xFFFF_FFFE and the Time Target Register (RTC_TTR) is reset to 0x0000_0000 at the assertion of the internal reset signal. At this time, the HIB_MODE signal is asserted to a logic 1 indicating that the MPC5125 is not in the hibernate mode. Because the time target register is initialized to 0x0000_0000, the actual time count register is always greater than or equal the time target register as long as the RTC_TTR remains at its initialized value of 0x0000_0000. In this case, the $\overline{\text{HIB\_MODE}}$ pin remains asserted to a logic 1. Thus, from the assertion of the internal reset signal which is caused by the application of power to V_{BAT}, the RTC is in an active state and the $\overline{\text{HIB\_MODE}}$ pin is asserted to a logic 1 and it remains asserted until the CPU writes a non-zero value to the time target register.

In an actual application, the $\overline{\text{HIB\_MODE}}$ pin is used to turn external power regulators on and off the supply power to the MPC5125.

The assertion of the HIB_MODE pin does not affect the operation of the CPU. For instance, if the HIB_MODE pin is not connected to anything (e.g., an external power regulator), then neither the assertion of the HIB_MODE pin nor the mechanisms that cause the assertion of the HIB_MODE pin have any effect on the operation of the MPC5125.

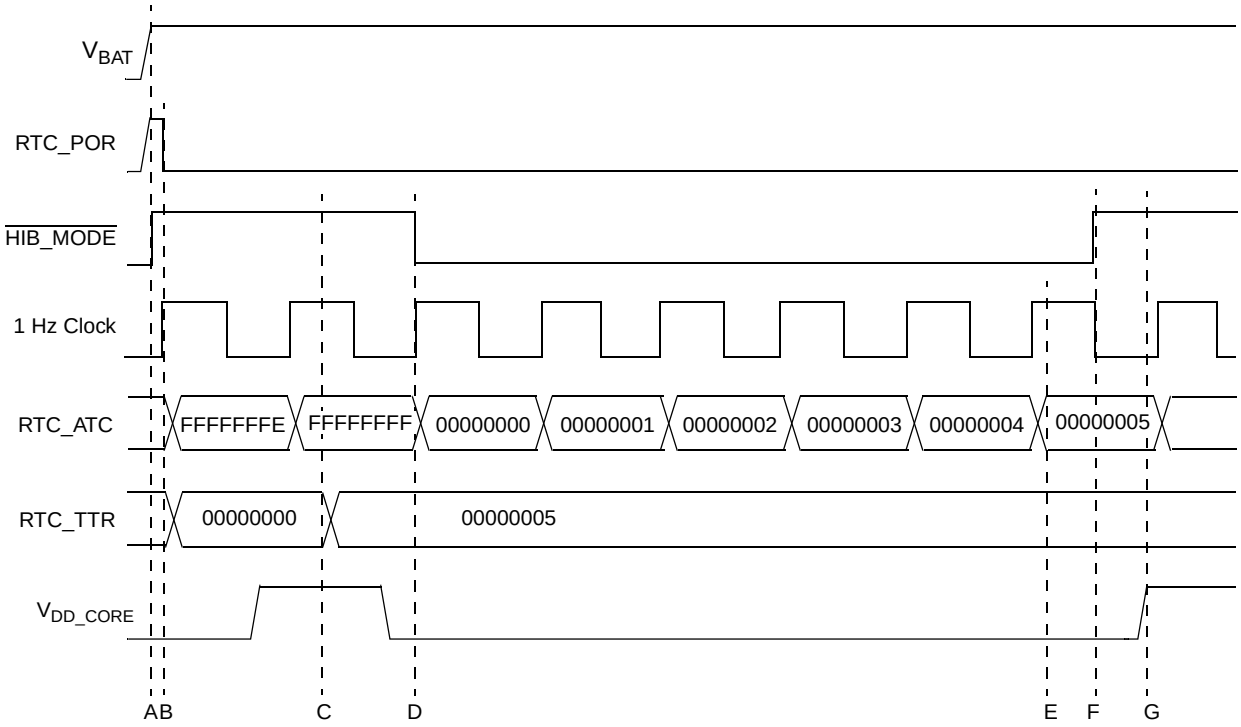


Figure 27-13. Timing Diagram of the RTC in Deep Sleep Mode

Point A Power is applied to the V_{BAT} pin. The application of power on the V_{BAT} pin caused the internal reset signal to assert to a logic 0 at point A. When the internal reset signal is asserted, the actual time count register is initialized to 0xFFFF_FFFE and the target time register is initialized to 0x0000_0000. Until these registers are modified with user software, the target time register is always less than or equal to the actual time count register. Under this condition, the HIB_MODE pin sets to a logic 1. If the HIB_MODE pin is connected to an external power regulator, the regulator is turned on, thus supplying power to the MPC5125.

Point B The internal RTC_POR signal clears to a logic 0, which negates the internal reset and enables the RTC. When the internal RTC_POR signal is released, the actual time count register is initialized to 0xFFFF_FFFE and the target time register is initialized to 0x0000_0000. Until these registers are modified with user software. The target time register is less than or equal to the actual time count register. Under this condition, the HIB_MODE pin sets to a logic 1. If the HIB_MODE pin is connected to an external power regulator, the regulator is turned on, thus supplying power to the MPC5125.

Point C The CPU writes 0x0000_0005 to the time target register and enables HIB_MODE function by clearing the DIS_HIB_MODE bit to zero.

Point D The time target register value is greater than the actual time count register. In the diagrams, however, the RTC_TTR value is 0x5, whereas the ACT value is 0xFFFF_FFFF. The is means the

ACT is greater than the RTC_TTR. This transfer occurs on the next falling edge of the internal 1 Hz clock signal which increments the RTC_ATC value. Because the time target register value is greater than the actual time count register value, the $\overline{\text{HIB\_MODE}}$ pin is driven to a logic 0. If the $\overline{\text{HIB\_MODE}}$ pin is connected to an external power regulator, the regulator is turned off, thus turning off power to the MPC5125. V_{DD_CORE} is turned off in response to the $\overline{\text{HIB\_MODE}}$ pin being driven to a logic 0.

Point E The actual time count register increments to 0x0000_0005 and now matches the contents of the target time register.

Point F The $\overline{\text{HIB\_MODE}}$ pin is driven to a logic 1 on the next negative edge of the 1 Hz

Point G V_{DD_CORE} is turned on in response to the $\overline{\text{HIB\_MODE}}$ pin being driven to a logic 1.

NOTE

This scenario assumes the wake up input sources are in an inactive state or disabled by the corresponding WU_SRC_EN bit.

27.4.4 RTC Response to Target Time Register/Actual Time Count Register and External Wakeup Sources

The timing diagram in Figure 27-14 shows the functionality of the RTC_TTR register and the functionality of detecting the positive edge of wakeup source, GPIO01. The RTC_TTR register is originally reset to 0x0000_0000. When RTC_TTR is updated to 0x0000_0005, then RTC_ATC < RTC_TTR and $\overline{\text{HIB\_MODE}}$ is asserted at the negative edge of the 1 Hz clock.

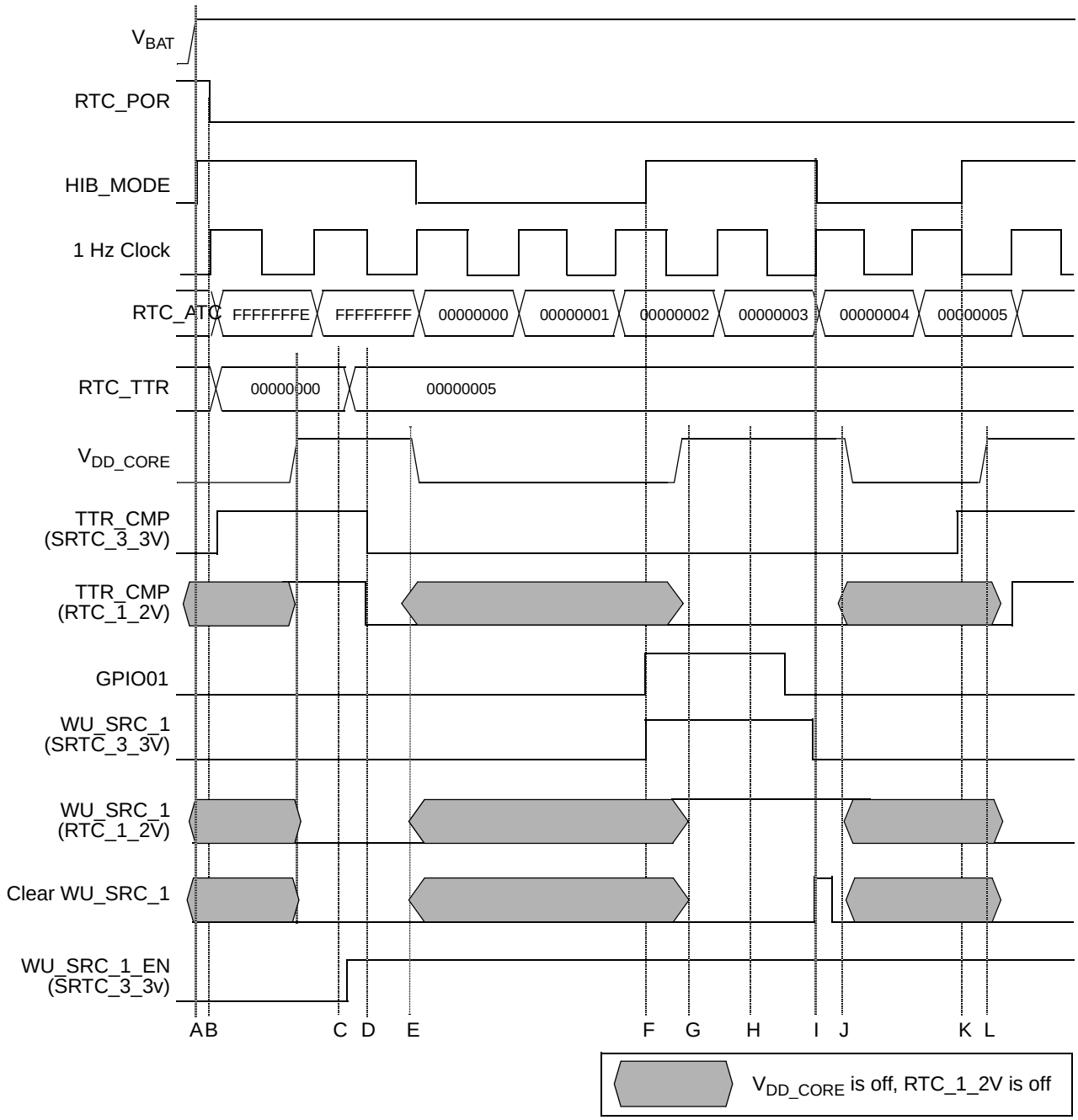


Figure 27-14. Timing Diagram of the RTC_TTR and the Positive Edge of GPIO01

The following is an explanation of Figure 27-14 timing diagram illustrating the functionality of the RTC_TTR register and the functionality of detecting the positive edge of wakeup source, GPIO01.

Point A Power is applied to the V_{BAT} pin. The application of power on V_{BAT} causes the internal RTC_POR pin to clear to a logic 0 at Point B.

Point B The internal RTC_POR signal asserts to a logic 0, which enables the real time clock. When the internal RTC_POR signal is released, the actual time count register is initialized to 0xFFFF_FFFE and the target time register is initialized to 0x0000_0000. Until these registers are

modified with user software. the target time register is less than or equal to the actual time count register. Under this condition, the HIB_MODE pin sets to a logic 1. If the HIB_MODE pin is connected to an external power regulator, the regulator is turned on, thus supplying power to the MPC5125.

Point C The CPU writes 0x0000_0005 to the time target shadow register. Next, the CPU writes to the RTC keep alive register and enables external wakeup source 1. Then, it enables HIB_MODE function by clearing the DIS_HIB_MODE bit to zero.

Point D Now, the time target register is greater than the actual time count register. This transfer occurs on the next falling edge of the internal 1 Hz clock signal which increments the RTC_ATC value. Because the time target register value is greater than the actual time count register value, the HIB_MODE pin is driven to a logic 0. If the HIB_MODE pin is connected to an external power regulator, the regulator is turned off, thus turning off power to the MPC5125.

Point E V_{DD_CORE} is turned off in response to the HIB_MODE pin being driven to a logic 0.

Point F The GPIO01 is asserted which causes the WU_SRC_1 bit in the RTC Keep Alive Register (RTC_KAR) to assert. This bit remains asserted, regardless of the logic level on GPIO01, until software clears the bit by writing a 1 to the WU_SRC_1 bit position. The assertion of GPIO01 causes the HIB_MODE pin to set to a logic 1, thus turning on the external power regulators to the MPC5125.

Point G The external voltage regulators turn on and apply power to the MPC5125.

Point H The CPU writes a 1 to the WU_SRC_1 bit in the RTC keep alive register to negate this bit.

Point I An internal clear signal clears the WU_SRC_1 bit in the RTC keep alive register. At this time, the WU_SRC_1_EN bit is set, thus enabling GPIO01 events to wakeup the device. Also, the actual time count register is less than the target time register. Under these conditions, the HIB_MODE pin clears to a logic 0, thus turning off the external power regulators to the device.

CAUTION

It is important to keep track of the values in the target time register and the actual time count register when enabling and disabling the external wakeup sources to keep from inadvertently clearing the HIB_MODE pin to a logic 0 and thus turning off the power to the MPC5125.

Point J The external voltage regulators turn off and power is removed from the MPC5125.

Point K The actual time count register has incremented to 0x0000_0005 and is equal to the target time register. On the next negative edge of the 1 kHz clock after the RTC_ATC value is equal to the RTC_TTR value, the HIB_MODE pin sets to a logic 1 causing the external power regulators to turn on.

Point L The external voltage regulators turn on and apply power to the MPC5125.

27.4.5 RTC Response to External Wakeup Sources

The following is an explanation of Figure 27-15 timing diagram illustrating the functionality of detecting GPIO[01:03] with WU_SRC_MODE disabled.

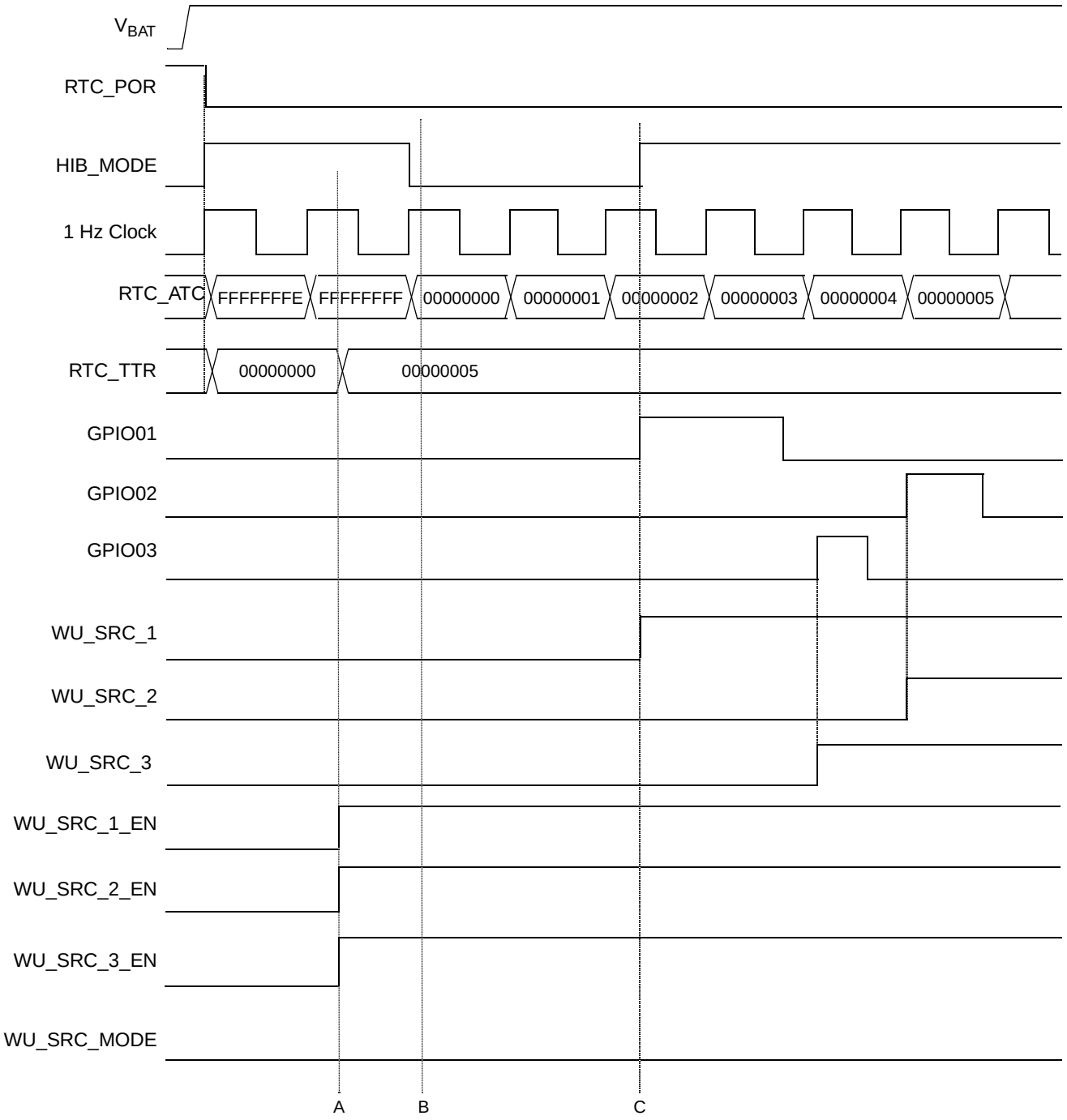


Figure 27-15. Timing Diagram Showing the Functionality of Detecting GPIO[01:03] with WU_SRC_MODE Disabled

Point A The $\overline{\text{HIB\_MODE}}$ signal is asserted in response to the release of the internal reset signal. The WU_SRC_1_EN, WU_SRC_2_EN and WU_SRC_3_EN bits are set in the RTC keep alive register to enable the respective external wakeup sources.

Point B The target time register is written to 0x0000_0005 and is greater than the RTC_ATC register. This causes the $\overline{\text{HIB\_MODE}}$ pin to clear to a logic 0.

Point C The GPIO01 pin sets to a logic 1 and causes the $\overline{\text{HIB\_MODE}}$ pin to set to a logic 1. This causes the external power regulators to turn on and apply power to the MPC5125.

27.4.5.1 Behavior of the RTC keep alive register WU_SRC_*x* bits when the WU_SRC_MODE is disabled

As shown in Figure 27-15, the WU_SRC_MODE is disabled. The $\overline{\text{HIB\_MODE}}$ pin is cleared to a logic 0 as soon as software writes the target time register to 0x0000_0005. The external RTC wakeup source pin, GPIO01, asserts. As soon as this pin asserts, the corresponding bit in the RTC keep alive register is set. Likewise, when GPIO02 and GPIO03 are set, their corresponding bits, WU_SRC_2 and WU_SRC_3 are also set. If the system software accesses the RTC Keep Alive Register (RTC_KAR) and more than one of the wakeup bits is set, it is not possible to see which one occurred first. On the other hand, any time that one of the six external RTC wakeup pins asserts, that occurrence is recorded in the RTC keep alive register. Recall that the WU_SRC_*x* bits are sticky bits and does not return to a logic 0 until a logic 1 is written to their respective bit position in the RTC keep alive register.

27.4.5.2 Behavior of the RTC keep alive register WU_SRC_*x* bits when the WU_SRC_MODE is enabled

As shown in Figure 27-16, the WU_SRC_MODE is enabled. The $\overline{\text{HIB\_MODE}}$ pin is cleared to a logic 0 as soon as software writes the target time register to 0x0000_0005. The external RTC wakeup source pin, GPIO01, asserts. As soon as this pin asserts, the corresponding bit in the RTC Keep Alive Register (RTC_KAR) is set; however, when GPIO02 and GPIO03 are set after GPIO01 is set and remains set, the WU_SRC_2 and WU_SRC_3 are not set. In this case, once an external RTC wakeup source is asserted and the corresponding bit in the RTC Keep Alive Register (RTC_KAR) is set, no more bits are set in the RTC keep alive register by asserting additional external RTC wakeup pins until the first external interrupt source is negated and its corresponding flag in the RTC Keep Alive Register (RTC_KAR) is also negated by writing a logic 1 to its bit position. This methodology allows the system software to determine which external wakeup source caused the device to exit the hibernate mode. After the bit in the RTC Keep Alive Register (RTC_KAR) is cleared that caused the device to exit the hibernate mode, then next external wakeup source to assert is recognized and then all further wakeup sources is inhibited until the current wakeup source is negated and its associated status bit is cleared. Recall that the WU_SRC_*x* bits are sticky bits and do not return to a logic 0 until a logic 1 is written to their respective bit position in the RTC Keep Alive Register (RTC_KAR).

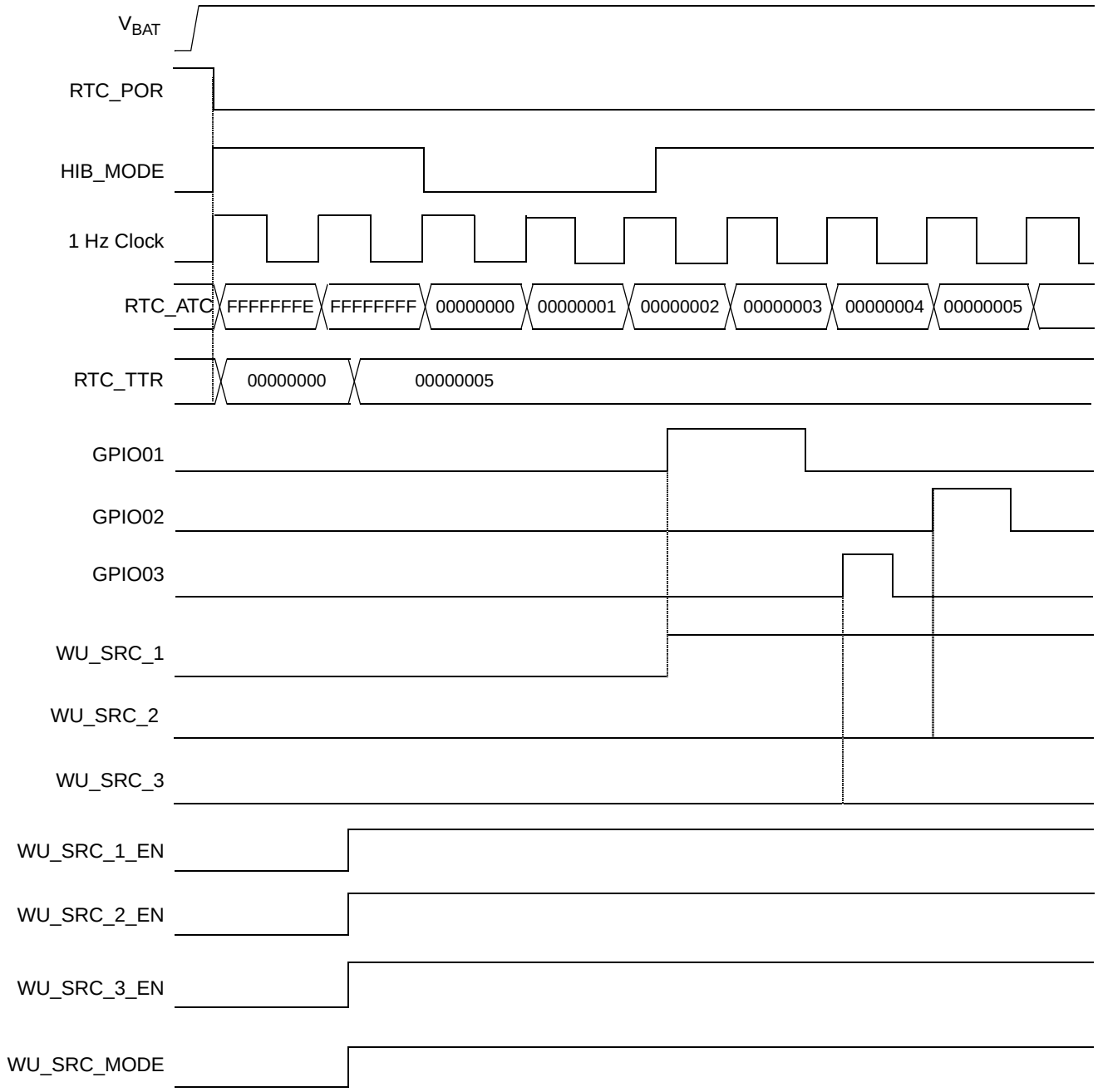


Figure 27-16. Timing Diagram Illustrating the Functionality of Detecting GPIO[01:03] with WU_SRC_MODE Enabled

Chapter 28
Secure Digital Host Controller (SDHC)

28.1 Introduction

The Secure Digital Host Controller module (SDHC) controls the Multimedia Card (MMC), Secure Digital (SD) memory card, and I/O cards by sending commands to cards and performing data accesses to/from the cards.

The MMC is a universal low-cost data storage and communication media that covers a wide area of applications such as electronic toys, organizers, PDAs, and smart phones. The MMC communication protocol is based on an advanced 6-pin serial bus that operates in a low-voltage range.

The SD is an evolution of MMC technology with two additional pins in the form factor. SD is specifically designed to meet the security, capacity, performance, and environment requirement inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment, and data transfer protocol are forward compatible with the MMC with some additions. The memory card invokes a copyright protection mechanism that complies with the SDMI security standard. The I/O card provides high-speed data I/O with low power consumption for mobile electronic devices.

The MMC/SD Host module integrates both MMC support along with SD memory and I/O functions.

Figure 28-1 is the system interconnection with this module.

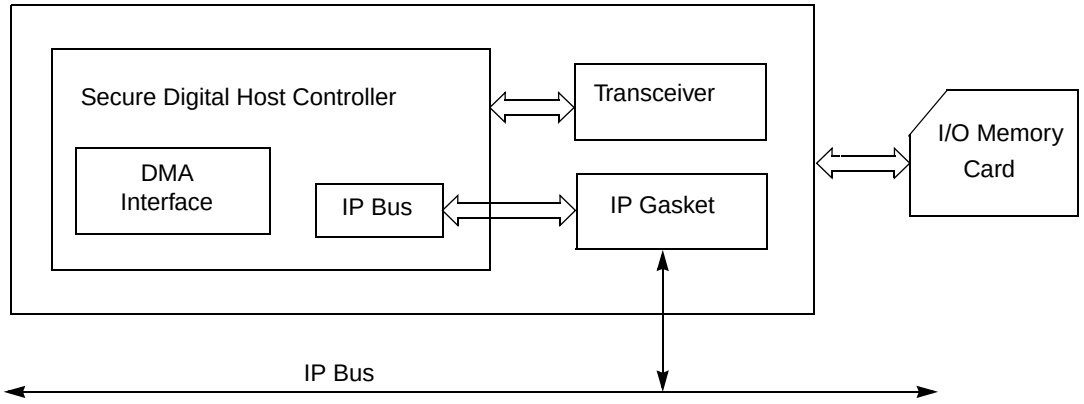


Figure 28-1. System Interconnection with the Secure Digital Host Controller

28.1.1 Features

The features of the SDHC module include:

- Fully compatible with the MMC System Specification Version 3.2, supports up to version 4.0

- Compatible with high speed MMC card of with using a 1-bit or 4-bit serial interface (8-bit interface not supported)
- Compatible with the SDIO Standard and SD Physical layer specification with one or four channel(s)
- Built-in programmable frequency counter for SDHC bus
- Maskable hardware interrupt for SDIO interrupt, internal status and FIFO status
- Built-in dual data FIFO buffer
- Plug and play (PnP) support
- Single/multi block access to the card including erase operation
- Multi-SD function support including multiple I/O and combined I/O and memory
- As many as six I/O functions plus one memory supported on single SD I/O card (Combo card)
- Support for interrupt via IRQ
- Support SDIO Interrupt detection during 1- or 4-bit access
- Block based data transfer between MMC card and SDHC (stream mode or SPI not supported)
- Block length of data transfer between Host and Card can be configured from 1–2048 bytes.

28.2 External Signal Description

Table 28-1. Signal Properties

Name	Port	Function	I/O	Reset	Pull up
MMC_SD_CLK	O	Clock for MMC/SD/SDIO card		0	
CMD	I/O	CMD line connect to card		1	Pull-up
DAT0	I/O	Data line both in 1-bit and 4-bit mode		1	Pull-up
DAT1	I/O	Data line or interrupt in 4-bit mode Interrupt in 1-bit mode		1	Pull-up
DAT2	I/O	Data line or Read wait in 4-bit mode Read wait in 1-bit mode		1	Pull-up
DAT3	I/O	Card Detect in Power up Data line in 4-bit mode Not used in 1-bit mode		0	Pull-down DAT3 if card detection needed through this bit, otherwise pull-up

28.2.1 Detailed Signal Descriptions

28.2.1.1 Overview

The SDHC has six external pins. MMC_SD_CLK is an internally generated clock used by the MMC/SD Card. The CMD I/O sends commands and receive responses from the card. Four data lines, DAT3~DAT0, perform data transfers between the host controller and the card.Table 28-2 shows the signal properties of these external pins.

Table 28-2 is a detailed description of the SDHC bus signals.

Table 28-2. Detailed Signal Descriptions

Signal	Descriptions
MMC_SD_CLK	This signal is the MMC/SD/SDIO card clock signal. The direction is from host to card. The frequency is referenced to the system clock.
CMD	This is a bidirectional signal. It is for card initialization and data transfer commands.
DAT0	This is a bidirectional signal for data transmission both in 1-bit and 4-bit mode.
DAT1	This is a bidirectional signal for data transmission or interrupt signal in 4-bit mode and is for interrupt signal when SDHC works in 1-bit mode.
DAT2	This is a bidirectional signal for data transmission or read wait in 4-bit mode and is for read/wait when SDHC works in 1-bit mode.
DAT3	This is a bidirectional signal for data transmission in 4-bit mode. This signal is not used in 1-bit mode. This signal is also used for card detect in both 4-bit and 1-bit modes.

28.3 Memory Map and Register Definition

The SDHC module contains 14 32-bit registers. This section provides the detailed descriptions for all SDHC registers. All registers must be accessed as 32-bit values. Byte/half word access is not allowed.

28.3.1 Memory Map

Table 28-3 shows the SDHC memory map. Write accesses to the reserved region are ignored. Read accesses to reserved locations return 0. To ensure compatibility with possible future revisions of this module, do not access reserved regions.

Table 28-3. SDHC memory map

Offset from SDHC_BASE[1] SDHC1: 0xFF40_1500 SDHC2: 0xFF40_5100	Register	Access[2]	Reset Value	Section/Page
0x00	SDHC_STR_STP_CLK—SDHC Clock Control Register	R/W	0x0000_0000	28.3.2.1/28-758
0x04	SDHC_STATUS—SDHC Status Register	R/W	0x0000_0000	28.3.2.2/28-759
0x08	SDHC_CLK_RATE—SDHC Card Clock Rate Register	R/W	0x0000_0008	28.3.2.3/28-763
0x0C	SDHC_CMD_DAT_CONT—SDHC Command and Data Control Register	R/W	0x0000_0000	28.3.2.4/28-764
0x10	SDHC_RES_TO—SDHC Response Time-out Register	R/W	0x0000_0040	28.3.2.5/28-766
0x14	SDHC_READ_TO—SDHC Read Time-out Register	R/W	0x0000_FFFF	28.3.2.6/28-767
0x18	SDHC_BLK_LEN—SDHC Block Length Register	R/W	0x0000_0000	28.3.2.7/28-768
0x1C	SDHC_NOB—SDHC Number of Block Register	R/W	0x0000_0000	28.3.2.8/28-769
0x20	SDHC_REV_NO—SDHC Revision Number Register	R	0x0000_0400	28.3.2.9/28-770

Table 28-3. SDHC memory map (Continued)

Offset from SDHC_BASE[1] SDHC1: 0xFF40_1500 SDHC2: 0xFF40_5100	Register	Access[2]	Reset Value	Section/Page
0x24	SDHC_INT_CNTR—SDHC Interrupt Control Register	R/W	0x0000_0000	28.3.2.10/28-771
0x28	SDHC_CMD—SDHC Command Number Register	R/W	0x0000_0000	28.3.2.11/28-774
0x2C	SDHC_ARG—SDHC Command Argument Register	R/W	0x0000_0000	28.3.2.12/28-775
0x30	Reserved			
0x34	SDHC_RES_FIFO—SDHC Command Response FIFO Access Register	R	0x0000_0000	28.3.2.13/28-775
0x38	SDHC_BUFFER_ACCESS—SDHC Data Buffer Access Register	R/W	0x0000_0000	28.3.2.14/28-776
0x3C–0xFF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

28.3.2 Register Descriptions

28.3.2.1 SDHC Clock Control (SDHC_STR_STP_CLK) Register

The SDHC Clock Control (SDHC_STR_STP_CLK) register allows software to reset the whole module and to enable or disable the MMC_SD_CLK to the card.

Figure 28-2 shows the SDHC_STR_STP_CLK register and Table 28-4 describes the bit fields.

Address: Base + 0x00 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	SLFCLR	0	0	0
W													SDHC RESET		START_CLK	STOP_CLK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-2. SDHC Clock Control (SDHC_STR_STP_CLK) Register

Table 28-4. SDHC_STR_STP_CLK field descriptions

Field	Description
SDHC RESET	SDHC Reset. Writes to the SDHC reset bit triggers the reset logic inside the SDHC. Reads from this bit always return 0. To reduce power consumption, the clock to the reset logic in the SDHC is off in normal operation. When there is one access to this register, the clock is enabled for one cycle. To complete the entire reset period, it needs at least 8 clock pulses to finish the reset cycle. To reset the SDHC module, it is recommended to write this register with value 0x0008, followed by 0x0009, and then 0x0001 eight times. Refer to Section 28.5.3.2, "Reset," for detailed information on software reset. 0 No effect. 1 Reset the SDHC module.
START_CLK	Start Clock. Writing a 1 to this bit starts the MMC_SD_CLK clock. Writing a value of 11 on Bits [1:0] of this register is not allowed. **Note:** The SDHC bus clock does not start immediately after writing to this bit. Poll the SDHC_STATUS[CARD_BUS_CLK_RUN] bit to ensure the SDHC clock is running. 0 No effect. 1 Starts the MMC/SD clock.
STOP_CLK	Stop Clock. Stops the MMC_SD_CLK clock when software writes a value of 1 to this bit. Software should not stop the MMC_SD_CLK during a transmission period. Writing a value of 11 to bits [1:0] of this register is not allowed. **Note:** A transmission period is the time from when a card data or access related command is submitted to the end of the access operation. **Note:** The SDHC bus clock does not stop immediately after writing to this bit. Poll the SDHC_STATUS[CARD_BUS_CLK_RUN] bit to ensure the SDHC clock is not running. 0 No effect. 1 Stops the MMC/SD clock.

28.3.2.2 SDHC Status (SDHC_STATUS) Register

The SDHC Status (SDHC_STATUS) register provides information about the status of SDHC operations, application FIFO status, error conditions, and interrupt status.

When the corresponding interrupt enable is enabled in the SDHC Interrupt Control (SDHC_INT_CNTR) register for any of these interrupts, the SDHC generates an interrupt request to the CPU. Software needs to clear the appropriate status bit to clear the corresponding interrupt. The interrupt status bits are cleared by using a write 1 to clear operation except for the data buffer ready status bits which can only be cleared by the read or write operation on the data buffer.

Figure 28-3 shows the SDHC Status Register and Table 28-5 describes the bit fields.

Address: Base + 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CARD_INSERTION	CARD_REMOVAL	YBUF_EMPTY	XBUF_EMPTY	YBUF_FULL	XBUF_FULL	BUF_UND_RUN	BUF_OVFL	0	0	0	0	0	0	0	0
W	w1c	w1c														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SDIO_INT_ACTIVE	END_CMD_RESP	WRITE_OP_DONE	READ_OP_DONE	WR_CRC_ERR_CODE		CARD_BUS_CLK_RUN	BUF_READ_RDY	BUF_WR_RDY	RESP_CRC_ERR	0	READ_CRC_ERR	WRITE_CRC_ERR	TIME_OUT_RESP	TIME_OUT_READ
W		w1c	w1c	w1c	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-3. SDHC Status (SDHC_STATUS) Register

Table 28-5. SDHC_STATUS field descriptions

Field	Description
CARD_INSERTION	Card Insertion. When this bit is set, the SDHC detects a value transition on the DAT[3:0] from b0111 to b1111. This can detect whether a card is inserted to the card socket based on DAT3 pull-up resistor of the card DAT3. When this bit is set, the SDHC generates an interrupt request if the card detection interrupt is enabled. This bit is read-only and can be cleared by writing 1 to this bit. 0 No card insertion detected. 1 Card insertion detected based on logic level changed on DAT3.
CARD_REMOVAL	Card Removal. When this bit is set, the SDHC detects a logic transition on the DAT[3:0] from b1111 to b0111. This can detect whether a card is removed from the card socket based on the DAT3 pull-up resistor of the card DAT3. When this bit is set, the SDHC generates an interrupt request if the card removal interrupt is enabled. This bit is read-only and can be cleared by writing a 1 to it. Software needs to clear this bit to clear the interrupt request from SDHC when card detection interrupt is enabled. 0 No card insertion detected. 1 Card removal detected based on logic level changed on DAT3.
YBUF_EMPTY	Y Data Buffer Empty. When this is set, it indicates the Y data buffer is empty during a write transfer. This bit is automatically cleared when the first byte of data is moved into the FIFO. Refer to Section 28.4.1, “Data Buffers,” for more information about the data buffers. 0 Y buffer is not empty. 1 Y buffer is empty.
XBUF_EMPTY	X Data Buffer Empty. When set, this bit indicates the X data buffer is empty during a write transfer. This bit is automatically cleared when the first byte of data is moved into the FIFO. Refer to Section 28.4.1, “Data Buffers,” for more information about the data buffers. 0 X buffer is not empty. 1 X buffer is empty.
YBUF_FULL	Y Data Buffer Full. When set, this bit indicates the Y data buffer is full during a read transfer. This bit is automatically cleared when the last byte of data is read out from the FIFO. Refer to Section 28.4.1, “Data Buffers,” for more information about the data buffers. 0 Y buffer is not full. 1 Y buffer is full.

Table 28-5. SDHC_STATUS field descriptions (Continued)

Field	Description
XBUF_FULL	X Data Buffer Full. When set, this bit indicates the X data buffer is full during a read transfer. This bit is automatically cleared when the last byte of data is read out from the FIFO. Refer to Section 28.4.1, "Data Buffers," for more information about the data buffers. 0 X buffer is not full. 1 X buffer is full.
BUF_UND_RUN	Buffer Underrun. When set, this bit indicates both X and Y data buffers are empty during a write transfer. In this case, the MMC_SD_CLK card clock is stopped automatically by hardware to wait for the DMA or CPU to put data into the buffers. An interrupt is triggered if the corresponding interrupt control bit is enabled. 0 No buffer underrun. 1 Buffer underrun during a write operation.
BUF_OVFL	Buffer Overflow. When set, this bit indicates both data buffers are full during a read operation. In this case, the MMC_SD_CLK card clock is stopped automatically by hardware to wait for the DMA or CPU to remove data out of one of the buffers. An interrupt is triggered if the corresponding interrupt control bit is enabled. Excess data is ignored by the SDHC. 0 No buffer overflow. 1 Buffer overflow during a read operation.
SDIO_INT_ ACTIVE	SDIO Interrupt Active. This bit indicates whether an interrupt from the SDIO card has been detected. When this bit is set, the SDHC generates an interrupt request if the SDIO interrupt is enabled. Software should clear the status bit to clear the interrupt request. However, a separate acknowledge command to the card may be required to clear the source of the SDIO interrupt. Writing a 1 to this bit clears it. 0 No interrupt detected. 1 Interrupt detected using SDIO card bus.
END_CMD_ RESP	End Command Response. This bit indicates whether a command was successfully transmitted to the card and the corresponding response was stored in the Response FIFO. This occurs after each command operation. When this bit is set, the SDHC generates an interrupt request if END_CMD_RESP interrupt is enabled . Software needs to clear this bit to clear the interrupt request. Writing a 1 to this bit clears it. 0 Command not successful, incomplete. or not applicable (no response). 1 Command transmitted successfully (response received). **Note:** When this bit is set, check the response stored in the response FIFO completed without fail. Also, check the RESP_CRC_ERR (Status[5]) and TIME_OUT_RESP(STATUS[1]) bits to determine if an error occurred.
WRITE_OP_ DONE	Write Operation Done. This indicates a write operation has completed. The flash card might need extra idle time for write accesses, which requires the SDHC module to wait until the card writes the buffered data to the inner flash memory. The WRITE_OP_DONE flag indicates the end of the write operation. When this bit is set, the pre-defined data bytes are written to the card. Software needs to send a STOP command to the card if the write command is a MMC/SD card write multi-block command. When this bit is set, SDHC generates an interrupt request if the WRITE_OP_DONE interrupt enable is enabled in the SCDH_INT_CNTR register. Software needs to clear this bit to clear the interrupt. This is accomplished by writing 1 to this bit. 0 Write operation in progress or incomplete. 1 Write operation complete. **Note:** When this bit is set, software also needs to check if the write operation completed without a cyclic redundancy check (CRC) error. Also, software needs to check the SDHC_STATUS[WR_CRC_ERR_CODE] bitfield and the SDHC_STATUS[WRITE_CRC_ERR] bit to determine if an error has occurred.

Table 28-5. SDHC_STATUS field descriptions (Continued)

Field	Description
READ_OP_ DONE	Read Operation Done. This bit is set at the end of a read operation. When this bit is set, pre-defined data bytes have been read from the card or a READ TIMEOUT has occurred. Software needs to send a STOP command to the card if the read command is a MMC or SD card read multi-block command. This bit can be cleared by writing 1 to it. When this bit is set, the SDHC generates an interrupt request if the READ_OP_DONE interrupt is enabled in the SDHC_INT_CNTR register. Software must clear this bit to clear the interrupt request. 0 Read operation is in progress or incomplete. 1 Read operation is complete. **Note:** When this bit is set, software also needs to check if the read operation completed without error. Also, software needs to check the SDHC_STATUS[READ_CRC_ERR] and SDHC_STATUS[TIME_OUT_READ] bits to determine if an error has occurred.
WR_CRC_ ERROR_CODE	Write CRC Error Code. This indicates CRC results from the card at the end of write operations. After receiving a block of data, the card checks the CRC bit and sends the CRC status. These two bits reflect the CRC status of the recent written data. If the card returns a negative CRC status, data is not written to the card. These two bits can be cleared by writing a value of 0b11 to them. 00 No transmission error, CRC status is 010 (positive). 01 Transmission error, CRC status is 110 (negative). 10 No CRC response. 11 Reserved. **Note:** The bits are valid only when the SDHC_STATUS[WRITE_CRC_ERR] bit is set.
CARD_BUS_ CLK_RUN	Card Bus Clock Run. This indicates whether the MMC_SD_CLK clock to the card is running. The clock rate setting and system configuration can be modified when the clock is turned off by setting the STOP_CLK bit in SDHC_STR_STP_CLK register. This bit can be cleared only after writing 1 to SDHC_STR_STP_CLK[STOP_CLK] to stop MMC_SD_CLK. 0 MMC/SD clock is stopped. 1 MMC/SD clock is running. Polling needs to be done on this bit to assure the SDHC clock is running or stopped.
BUF_READ_ RDY	Buffer Read Ready. This status is set if a buffer (either X buffer or Y buffer) is full during read operations. An interrupt is triggered for non-DMA transfers if the SDHC_INT_CNTR[BUF_READ_EN] bit is set, or a DMA request is asserted for DMA transfers. 0 Not ready to read buffer. 1 Ready to read buffer.
BUF_WR_ RDY	Buffer Write Ready. This status is set if a buffer (either X buffer or Y buffer) is available during write operations. An interrupt is triggered for non-DMA transfers if the SDHC_INT_CNTR[BUF_WRITE_EN] bit is set or a DMA request is asserted for DMA transfers. This bit is only set when the SDHC is performing write operation to the card. 0 Not ready to write buffer. 1 Ready to write buffer.
RESP_CRC_ ERR	Response CRC Error. This indicates a transmission error occurred on the SD_CMD line during a response transfer. Writing 1 to this bit clears this bit. 0 No error. 1 Response CRC error occurred.
READ_CRC_ ERR	Read CRC Error. This indicates a transmission error occurred on the DAT line during a card read. Software should retry the transmission. Writing a 1 to this bit clears the bit. 0 No error. 1 CRC read error occurred.
WRITE_CRC_ ERR	Write CRC Error. This indicates a transmission error occurred on the DAT line during a card write operation. Software needs to check the SDHC_STATUS[WR_CRC_ERR_CODE] bitfield for more information about the CRC error. Writing a 1 to this bit clears this bit. 0 No error. 1 CRC write error occurred.

Table 28-5. SDHC_STATUS field descriptions (Continued)

Field	Description
TIME_OUT_RESP	Time Out Response. This indicates command response was not received in time specified in the SDHC_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 had already 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 Writing a 1 to this bit clears this condition. 0 No error. 1 Time out response error occurred.
TIME_OUT_READ	Time Out Read. Indicates expected data from the card was not received in time specified in the SDHC_READ_TO register. The TIME_OUT_READ bit is cleared by an internal status change or by removing the source of the error. Writing a 1 to this bit clears this bit. 0 No error. 1 Time out read data error occurred.

28.3.2.3 SDHC Clock Rate (SDHC_CLK_RATE) Register

Refer to Section 28.4.7, “System Clock Controller,” for the clock scheme.

The high frequency input clock, SDHC_CLK, derives the low frequency clock (CLK_20M) the card uses. The divide circuitry consists of a 4-bit divider followed by a 12-bit prescaler. The SDHC_CLK is first divided by the 4-bit divider to derive the signal, CLK_DIV. The 12-bit prescaler divides the CLK_DIV to derive CLK_20M, which the card can use.

CLK_20M is used internally by the SDHC.

Figure 28-4 shows the SDHC_CLK_RATE register and Table 28-6 describes the bit fields.

Address: Base + 0x08 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CLK_PRESCALER												CLK_DIVIDER			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 28-4. SDHC Clock Rate (SDHC_CLK_RATE) Register

Table 28-6. SDHC_CLK_RATE field descriptions

Field	Description
CLK_ PRESCALER	Clock Prescaler. Specifies divider value to generate CLK_20M from CLK_DIV. 0x000 CLK_20M is CLK_DIV 0x001 CLK_20M is CLK_DIV/2 0x002 CLK_20M is CLK_DIV/4 0x004 CLK_20M is CLK_DIV/8 0x008 CLK_20M is CLK_DIV/16 0x010 CLK_20M is CLK_DIV/32 0x020 CLK_20M is CLK_DIV/64 0x040 CLK_20M is CLK_DIV/128 0x080 CLK_20M is CLK_DIV/256 0x100 CLK_20M is CLK_DIV/512 0x200 CLK_20M is CLK_DIV/1024 0x400 CLK_20M is CLK_DIV/2048 0x800 CLK_20M is CLK_DIV/4096
CLK_DIVIDER	Clock Divider. Specifies the divider value to generate CLK_DIV from input clock SDHC_CLK. 0x0 CLK_DIV is SDHC_CLK 0x1 CLK_DIV is SDHC_CLK/2 0x2 CLK_DIV is SDHC_CLK/3 0x3 CLK_DIV is SDHC_CLK /4 0x4 CLK_DIV is SDHC_CLK /5 0x5 CLK_DIV is SDHC_CLK /6 0x6 CLK_DIV is SDHC_CLK /7 0x7 CLK_DIV is SDHC_CLK /8 0x8 CLK_DIV is SDHC_CLK /9 0x9 CLK_DIV is SDHC_CLK /10 0xA CLK_DIV is SDHC_CLK /11 0xB CLK_DIV is SDHC_CLK /12 0xC CLK_DIV is SDHC_CLK /13 0xD CLK_DIV is SDHC_CLK /14 0xE CLK_DIV is SDHC_CLK /15 0xF CLK_DIV is SDHC_CLK /16

NOTE

Maximum frequency of MMC_SD_CLK is SDHC_CLK/1 when CLK_DIVIDER and CLK_PRESCALER are set to 0x0.

28.3.2.4 SDHC Command and Data Control (SDHC_CMD_DAT_CONT) Register

The SDHC Command and Data Control (SDHC_CMD_DAT_CONT) register allows specifying the format of data and response, and controls the Read/Wait cycle. After configuring this register, enabling the MMC_SD_CLK causes the command and argument configured in the Command Number (SDHC_CMD) register and the Command Argument (SDHC_ARG) register to be sent out to the card.

Figure 28-5 shows the SDHC_CMD_DAT_CONT register and Table 28-7 describes the bit fields.

Address: Base + 0x0C

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMD_RESUME	0	0	CMD_RESP_LONG_OFF	STOP_READWAIT	START_READWAIT	BUS_WIDTH		INIT	0	0	WRITE_READ	DATA_ENABLE	FORMAT_OF_RESPONSE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-5. SDHC Command and Data Control (SDHC_CMD_DAT_CONT) Register

Table 28-7. SDHC_CMD_DAT_CONT field descriptions

Field	Description
CMD_RESUME	Command Resume. Restores the SDHC_CMD_DAT_CONT register after READ/WAIT cycle for SDIO card. 0 Issues command to card. 1 Does not issue command to card.
CMD_RESP_LONG_OFF	Command Response Long Off. Allows STATUS[13] END_CMD_RESP bit to be self-cleared when condition to generate this bit disappears. This is used in the Read/Wait cycle. For SD/MMC operation, keep this bit at 0. 0 Bit not cleared when read. 1 Allows bit clearance.
STOP_READWAIT	Stop Read/Wait. Ends the Read/Wait cycle for SDIO. When this bit is set, SDHC does not drive DAT2 output low so the SDIO card would end the Read/Wait cycle. For operation of SD/MMC, keep this bit at 0. 0 No effect. 1 Ends Read/Wait cycle.
START_READWAIT	Start Read/Wait. Starts the Read/Wait cycle for SDIO. When this bit is set, SDHC makes the DAT2 output low and forces the SDIO card to enter READWAIT cycle. For SD/MMC operation, keep this bit at 0. 0 No effect. 1 Starts Read/Wait cycle.
BUS_WIDTH	Bus Width. Specifies the width of the data bus. These two bits must be set according to current card bit mode. 00 1-bit. 01 Reserved. 10 4-bit. 11 Reserved.

Table 28-7. SDHC_CMD_DAT_CONT field descriptions (Continued)

Field	Description
INIT	Initialize. Specifies whether the additional 80-clock (MMC_SD_CLK) cycle prefix (to initialize the card) occurs before every command. INIT enables/disables the additional 80-clock initialization time. 0 Disable 80 initialization clocks. 1 Enable 80 initialization clocks.
WRITE_READ	Write/Read. Specifies whether the data transfer of current command is a write or read operation 0 Read. 1 Write.
DATA_ENABLE	Data Enable. Specifies whether the current command includes a data transfer. 0 No data transfer included. 1 Date transfer include.
FORMAT_OF_ RESPONSE	Format of Response. Sets the expected response format for current command. Refer to the SD I/O Specification 1.0 for detail information of the response format. 000 No response for Current command. 001 Format R1/R5/R6 (48-bit Response with CRC7). 010 Format R2 (136-bit, CSD/CID read response). 011 Format R3/R4(48-bit Response without CRC check). 1xx Reserved.

28.3.2.5 SDHC Response Time Out (SDHC_RES_TO) Register

The SDHC Response Time Out (SDHC_RES_TO) register defines an interval within which a response must be returned or a timeout error occurs. After SDHC sends out a command, if the card does not respond within the specified interval, the STATUS[TIME_OUT_RESP] and the STATUS[END_CMD_RESP] status bits are set.

Figure 28-6 shows the SDHC_RES_TO register and Table 28-8 describes its bit fields.

Address: Base + 0x10 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RESPONSE TIME OUT							
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 28-6. SDHC Response Time Out (SDHC_RES_TO) Register

Table 28-8. SDHC_RES_TO field descriptions

Field	Description
RESPONSE TIME OUT	Response Timeout. This value determines the interval which detects response timeout. The clock starts counting when the last bit of the command is sent. The clock counts unit is MMC_SD_CLK to card. 0x01 1 clock count 0x02 2 clock counts ... 0x40 64 clock counts (default) ... 0xFF 255 clock counts

28.3.2.6 SDHC Read Time Out (SDHC_READ_TO) Register

The SDHC Read Time Out (SDHC_READ_TO) register defines an interval that read data must be returned within or a timeout error occurs. After the SDHC sends out data read command, if no data is returned within the specified interval, the STATUS[TIME_OUT_READ] and STATUS[READ_OP_DONE] status bits are set.

Figure 28-7 shows the SDHC_READ_TO register and Table 28-9 describes the bit fields.

Address: Base + 0x14 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA_READ_TIME_OUT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 28-7. SDHC Read Time Out (SDHC_READ_TO) Register

Table 28-9. SDHC_READ_TO field descriptions

Field	Description
DATA_READ_ TIME_OUT	Data Read Timeout. This value determines the interval read data timeouts are detected. Check the timeout limit of the card and clock frequency to configure this register. For safety, 0xFFFF is recommended. The time-out clock starts counting when the last bit of the command is sent. The count unit is MMC_SD_CLK/256. The maximum delay the SDHC can tolerate for a data time out relates to the card clock. If the clock is 25 MHz and register is 0xFFFF, the maximum delay the SDHC waits is about 670 ms. If the card does not give the data in 670 ms, the SDHC issues a data read time out error and terminates current data read operation. This meets the SD physical layer specification, with a typical time out limit of 100 to 200 ms. However, for some SDIO cards, the time out limit may be as long as one second. In such cases, lower the MMC_SD_CLK frequency to accommodate the delay to 1 s. In this case, software needs to configure the MMC_SD_CLK to about 16 MHz and set this register to 0xFFFF.

28.3.2.7 SDHC Block Length (SDHC_BLK_LEN) Register

The SDHC Block Length (SDHC_BLK_LEN) register defines the number of bytes in a block. Because the stream mode of MMC is not supported, block length must be set for every transfer. Block lengths supported by the SDHC range from 1 to 2048 bytes, but the block size supported by the card must be checked before configuring this register. For SDIO, the block length must be less than the maximum block size defined in the card's CCCR. For SD/MMC, the block length must be less than the maximum block size defined in the card's CSD register.

NOTE

Software should write to this register only when no SD bus transaction is executing.

Figure 28-8 shows the SDHC_BLK_LEN register and Table 28-10 describes the bit fields.

Address: Base + 0x18 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	BLOCK LENGTH											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-8. SDHC Block Length (SDHC_BLK_LEN) Register

Table 28-10. SDHC_BLK_LEN field descriptions

Field	Description
BLOCK LENGTH	Block Length. Specifies the number of bytes in a block during data transfer (block size). For MMC and SD cards, the value set must remain the same as the Balkan set in the CARD. For SDIO, IO access is performed through the CMD53 IO_RW_EXTEND command. Command has two modes: • Byte mode. For byte mode, its operation is similar to a single block transfer command for SD where the block length is the byte count in the command argument. • Block mode. For block mode, its operation is similar to a multi-block transfer command for the SD where the block length is the block size defined in the command argument. For multi-block data transfers, a block length equal to an integer multiple of the data buffer size is preferred. Otherwise, buffer utilization is poor. If the data size that needs to be transferred is not an integer multiple of the buffer size, there are two options to transfer the data: • Option 1: Split the transaction. The remainder of block size data is transferred by using a single block command for the last transfer. • Option 2: Add filler data in the last block to fill the block size to be as large as the buffer size. The data buffer size is 64 bytes in 4-bit mode and 16 bytes in 1-bit mode. Refer to Section 28.4.1, “Data Buffers,” for more information about data buffer. 0x000 0 byte 0x001 1 byte ... ... 0x7FF 2047 bytes 0x800 2048 bytes 0x801~0xFFF Reserved

28.3.2.8 SDHC Number of Blocks (SDHC_NOB) Register

The SDHC Number of Blocks (SDHC_NOB) register defines the number of blocks in the block transfer mode. This register and the Block Length Register determines the number of bytes to be transferred during one command. The number decrements every time a block transfer completes and stops when the count reaches zero. When all data transfers are completed, the STATUS[READ_OP_DONE] is set if it is a read (from card) transfer, or the STATUS[WRITE_OP_DONE] is set if it is a write (to card) transfer.

Software should write to this register only when no MMC/SD transaction is executing.

The maximum data size to be transferred in bytes equals the block length multiplied by the number of blocks.

Figure 28-9 shows the SDHC_NOB register and Table 28-11 describes the bit fields.

Address: Base + 0x1C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NOB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-9. SDHC Number of Blocks (SDHC_NOB) Register

Table 28-11. SDHC_NOB field descriptions

Field	Description
NOB	Specifies the number of blocks in a block transfer. One block should be set if the data transfer command is a single block transfer command or IO_RW_EXTEND (CMD53) in byte mode. For multi-block transfer command to SD/MMC card and IO_RW_EXTEND (CMD53) in block mode to SDIO card, this register should be set the block count software expects. Number of blocks can range from 0 to 65535. For SD Memory card or a memory part of an SDIO combo card, send CMD12 to stop the multi-block transfer. For an SDIO CMD53 in block mode when software needs to abort the transfer earlier, use CMD52 IO-Abort to abort the transfer. 0x0000 0 block 0x0001 1 block ... ... 0xFFFF 65535 blocks **Note:** The maximum transfer blocks is 65535. If software uses an infinite transfer command to transfer data, such as the multi-block transfer command for memory card or the infinite block transfer CMD53 for SDIO card, this register needs to be set to the real number of blocks expected to be transferred. Also, to abort the transfer,the CMD12 or CMD52 IO-Abort must be used.

28.3.2.9 SDHC Revision Number (SDHC_REV_NO) Register

The SDHC Revision Number (SDHC_REV_NO) register is a read-only register displaying the revision number of the module.

Figure 28-10 shows the SDHC_REV_NO register and Table 28-12 describes the bit field.

Address: Base + 0x20 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Revision Number[15:0]															
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 28-10. SDHC Revision Number (SDHC_REV_NO) Register

Table 28-12. SDHC_REV_NO field descriptions

Field	Description
REVISION NUMBER	Revision Number. Specifies revision number of the MMC/SD module. This is fixed at 0x0000_0400.

28.3.2.10 SDHC Interrupt Control (SDHC_INT_CNTR) Register

When certain events occur in the module, the SDHC has the ability to set an interrupt as well as set corresponding status register bits. The SDHC Interrupt Control (SDHC_INT_CNTR) register allows control over whether these interrupts should be recognized. Interrupts are ORed to provide a single interrupt IPI_IRQ to the system. Software must read the status to determine the source of the event.

Figure 28-11 shows the SDHC_INT_CNTR register and Table 28-13 describes the bit fields.

Address: Base + 0x24 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CARD_INSERTION_EN	CARD_REMOVAL_EN	SDIO_IRQ_EN	DAT0_EN	0	0	0	0	0	0	0	BUF_READ_EN	BUF_WRITE_EN	END_CMD_RES	WRITE_OP_DONE	READ_OP_DONE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-11. SDHC Interrupt Control (SDHC_INT_CNTR) Register

Table 28-13. SDHC_INT_CNTR field descriptions

Field	Description
CARD_ INSERTION_EN	Card Insertion Enable. Setting this bit enables the card insertion interrupt. Because card detection is through the value of DAT3 data line, if this card is in 4-bit mode, any data transfers in the DAT3 line causes false card insertion interrupts to be generated. Card insertion interrupt should be disabled after the first time card insertion is detected. To avoid false status bit generation during data transfer, card insertion status is masked by this bit. It should only be enabled after the card is removed from the socket. The default of this bit is to disable the card insertion interrupt. When this interrupt is detected, write a 1 to the STATUS[31] bit to clear the card insertion status interrupt. 0 Card insertion interrupt disabled. 1 Card insertion interrupt enabled.
CARD_ REMOVAL_EN	Card Removal Enable. Setting this bit enables the card removal interrupt. Because card detection is through the value of the DAT3 data line, if this card is in 4-bit mode, the data transfer through the DAT3 line causes false card removal interrupt to be generated. The card removal interrupt should only be enabled when no active data transfers exist on the DAT3 line. To avoid the false status bit generation during data transfer, the card insertion status is masked by this bit. The default of this bit is to disable the card removal interrupt. When this interrupt is detected, write a 1 to the STATUS[30] bit to clear the card removal status interrupt. 0 Card removal interrupt disabled. 1 Card removal interrupt enabled. **Note:** SDHC uses IPS_CLK to detect the SDIO card interrupt wakeup event when this bit is set.
SDIO_IRQ_EN	SDIO Interrupt Enable. Masks the interrupt from the SD I/O card to the SDHC module interrupt. 0 SD I/O interrupt disabled. 1 SD I/O interrupt enabled. **Note:** SDHC uses IPS_CLK to detect the SDIO card interrupt wakeup event when this bit is set.
DAT0_EN	Data Enable. Identifies how the SD I/O interrupt is detected. An interrupt is determined by DAT 1 low, but this bit is an optional setting for the SDIO bit. When SDHC preforms data transfer and the SD bus mode is 1-bit mode, set this bit to 0. 0 SD I/O's Interrupt detection based on DAT[3:0] = b110*x*. 1 SD I/O's Interrupt detection based on DAT[3:0] = b1101.
BUF_READ_EN	Bus Read Enable. This bit controls the buffer read ready interrupt. If the bit is 1, the interrupt is enabled. When the buffer becomes full during a read operation, an interrupt is generated. Move the data out of the FIFO and clear the BUF_READ_RDY bit to clear the interrupt. 0 Buffer status interrupt disabled. 1 Buffer status interrupt enabled.

Table 28-13. SDHC_INT_CNTR field descriptions (Continued)

Field	Description
BUF_WRITE_EN	Bus Write Enable. This bit controls the buffer write ready interrupt. If this bit is 1, interrupt is enabled. When the buffer becomes empty during a write operation, an interrupt is generated. Write data to the FIFO and clear the STATUS[BUF_WR_RDY] bit to clear the interrupt. 0 Buffer status interrupt disabled. 1 Buffer status interrupt enabled.
END_CMD_RES	End Command Response. This bit controls the interrupt generation on the status at the end of the command response. When this bit is 1, the SDHC generates an interrupt at the end of the command response status. 0 End Command-response interrupt disabled. 1 End Command-response interrupt enabled.
WRITE_OP_ DONE	Write Operation Done. This bit controls the interrupt generation for the status of write operation. When the interrupt enabled, the SDHC generates an interrupt when the configured bytes of data are transferred to the card. 0 Write_OP_DONE interrupt disabled 1 Write_OP_DONE interrupt enabled
READ_OP_ DONE	Read Operation Done. This bit controls the interrupt generation for the status of read operation completion. When the interrupt is enabled, the SDHC generates an interrupt when the pre-defined bytes of data are transferred from the card. 0 READ_OP_DONE interrupt disabled 1 READ_OP_DONE interrupt enabled

When an interrupt is generated, there may be some error bits in the STATUS register set and the pending interrupt status (source of the interrupt must be cleared). Check the error status bit to make sure there is no error in the SDHC operation. For example, when the STATUS[READ_OP_DONE] status is set or the READ_OP_DONE interrupt is detected, check the STATUS[READ_CRC_ERR] and STATUS[TIME_OUT_READ] bits to make sure the read operation completed without a CRC error or a time out error. Another example is a write operation, if both the STATUS[WRITE_OP_DONE] and STATUS[WRITE_CRC_ERR2] bits are set. This means the write operation ended with CRC error. See Table 28-14 for a summary of the relationship between the interrupt, interrupt control register, and status registers in the SDHC.

Table 28-14. Interrupt Mechanisms

Source: SDHC_STATUS Bit	Does status directly generate interrupt?	SDHC_INT_CNTR Bit	Interrupt/Status Clear Method
TIME_OUT_READ	No, alert using the SDHC_STATUS[READ_OP_DONE] bit.	READ_OP_DONE	Clear status by writing 1
TIME_OUT_RESP	No, alert using the SDHC_STATUS[END_CMD_RESP] bit.	END_CMD_RES	Clear status by writing 1
WRITE_CRC_ERR	No, alert using the SDHC_STATUS[WRITE_OP_DONE] bit.	WRITE_OP_DONE	Clear status by writing 1
READ_CRC_ERR	No, alert using the SDHC_STATUS[READ_OP_DONE] bit.	READ_OP_DONE	Clear status by writing 1
RESP_CRC_ERR	No, alert using the using SDHC_STATUS[END_CMD_RESP] bit.	END_CMD_RES	Clear status by writing 1

Table 28-14. Interrupt Mechanisms (Continued)

Source: SDHC_STATUS Bit	Does status directly generate interrupt?	SDHC_INT_CNTR Bit	Interrupt/Status Clear Method
BUF_WR_RDY	Yes	BUF_WRITE_EN	Clear status by writing data to FIFO buffer
BUF_READ_RDY	Yes	BUF_READ_EN	Clear status by reading data from FIFO buffer
READ_OP_DONE	Yes	READ_OP_DONE	Clear status by writing 1
WRITE_OP_DONE	Yes	WRITE_OP_DONE	Clear status by writing 1
END_CMD_RESP	Yes	END_CMD_RESP	Clear status by writing 1
SDIO_INT_ACTIVE	Yes	SDIO_IRQ_EN	Clear status by writing 1
CARD_REMOVAL	Yes	CARD_REMOVAL_EN	Clear status by writing 1
CARD_INSERTION	Yes	CARD_INSERTION_EN	Clear status by writing 1

28.3.2.11 SDHC Command Number (SDHC_CMD) Register

The command to the SD card is always 48 bits long. It contains 1 start bit, 1 direction bit, 6 command number bits, 32 argument bits, 7 CRC bits, and 1 end bit. For more details on the format of the command, refer to the SD physical layer spec. Refer to related card specification for detailed information about each command.

The SDHC Command Number (SDHC_CMD) register automatically generates the command start bit, direction bit, CRC7 bits, and end bit in hardware. Configure only the SDHC_CMD register and SDHC Command Argument (SDHC_ARG) register to issue a command to the card.

Figure 28-12 shows the SDHC_CMD register and Table 28-15 describes the bit fields.

Address: Base + 0x28 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	COMMAND NUMBER					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-12. SDHC Command Number (SDHC_CMD) Register

Table 28-15. SDHC_CMD field descriptions

Field	Description
COMMAND NUMBER	Command Number. The SDHC module communicates with the MMC/SD/SDIO 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 SDHC CMD Argument (SDHC_ARG register). 0x00 CMD0 0x01 CMD1 ... ... 0x3F CMD63 **Note:** Check the detailed information from the related card specification.

28.3.2.12 SDHC Command Argument (SDHC_ARG) Register

The SDHC Command Argument (SDHC_ARG) register contains the MMC/SD/SDIO command argument.

Figure 28-13 shows the SDHC_ARGregister and Table 28-16 describes the bit fields.

Address: Base + 0x2C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ARG[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ARG[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-13. SDHC Command Argument (SDHC_ARG) Register

Table 28-16. SDHC_ARG field descriptions

Field	Description
ARG	Command Argument. Specifies the argument for the current command. **Note:** Check the detailed command argument information from the related card specification.

28.3.2.13 SDHC Response FIFO Access (SDHC_RES_FIFO) Register

The SDHC Response FIFO Access (SDHC_RES_FIFO) register is an 8 x 16 bit FIFO in the SDHC used to store the response from the card. The FIFO data can be read using this register. The most significant 16 bits of the response are accessed first, and the least significant 16 bits are accessed last.

Figure 28-14 shows the SDHC_RES_FIFO register and Table 28-17 describes the bit fields.

Address: Base + 0x34 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESPONSE_CONTENT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-14. SDHC Response FIFO Access (SDHC_RES_FIFO) Register

Table 28-17. SDHC_RES_FIFO field descriptions

Field	Description
RESPONSE_ CONTENT	Response Content FIFO access register. A FIFO in the SDHC stores the command response received from the card. Every time the Host sends a command to a card, the current contents stored in the FIFO are cleared and a new response argument is stored into the response FIFO. According to the SD card spec, command response size can be 48-bit or 136-bit (R2 response). Refer to the SD Memory Card Specification for more detailed information about the command response format. The response FIFO is 8 × 16 bits (128 bits). For a 48-bit response, only 48 bits of the FIFO have valid contents and software must perform three reads to this response FIFO access register to retrieve the entire 48-bit response content. For a 136-bit R2, response (response for CID[127:0] or CSD[127:0] register), only the contents of the 128-bit CID and CSD register are stored in the response FIFO. This first byte of the R2 response is not stored in the response FIFO. Retrieve the CIS/CSD register from the response FIFO through eight accesses to the FIFO access register. The CRC bit in the response is not stored in the response FIFO. This response FIFO is read only.

28.3.2.14 SDHC Data Buffer Access (SDHC_BUFFER_ACCESS) Register

The SDHC uses two 64-byte data buffers in an alternating manner to transfer data by the DMA and the SD card simultaneously to maximize throughput between the two clock domains (the IP peripheral clock, SDHC_CLK, and the host clock, CLK_20M). These buffers are temporary storage for data transferred between the host system and the card and vice versa. Read or write data to the buffers through this buffer access register. Refer to Section 28.4.1, “Data Buffers,” for more information about the data buffers.

In the read operation, the SDHC stores data received from the card into the buffer. Move the data out of the buffer when the buffer is full.

In the write operation, the SDHC fetches data from the buffer and transfers it to the card. Access the data buffer through the SDHC data buffer access register. Move data into the buffer when the buffer is empty.

Figure 28-15 shows the SDHC_BUFFER_ACCESS register and Table 28-15 describes the bit fields.

Address: Base + 0x38 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FIFO CONTENT[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	FIFO CONTENT[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-15. SDHC Data Buffer Access (SDHC_BUFFER_ACCESS) Register

Table 28-18. SDHC_BUFFER_ACCESS field descriptions

Field	Description
FIFO CONTENT	FIFO Content. These bits hold 32-bit data upon a read or write transfer. The size of the FIFO is 4 × 32 bits (16 bytes in total) for SD 1-bit mode and 16 × 32 bits (64 bytes in total) for SD 4-bit mode. For reception, the SDHC controller generates a DMA request when the FIFO is full. Upon receiving this request, the DMA starts transferring data from the SDHC FIFO to system memory by reading the data buffer access register for a number of pre-defined bytes. For transmit, the SDHC controller generates a DMA request when the FIFO is empty. Upon receiving this request, the DMA starts moving data from the system memory to the SDHC FIFO by writing to the data buffer access register for a number of pre-defined bytes.

28.4 Functional Description

The SDHC module controls the MMC, SD memory card, and I/O cards by sending commands to cards and performing data accesses to/from the cards.

The following sections provide a brief functional description of the major system blocks, including the DMA interface, memory controller, logic/command controller, and system clock controller.

28.4.1 Data Buffers

The SDHC uses two data buffers in an alternating manner to transfer data through the DMA and the SD card simultaneously to maximize throughput between the two clock domains (the IP peripheral clock, IPG_PERCLCK, and the host clock, CLK_20M). See Figure 28-16 for illustration of the buffering scheme. These buffers are used as temporary storage for data transferred between the host system and the card.

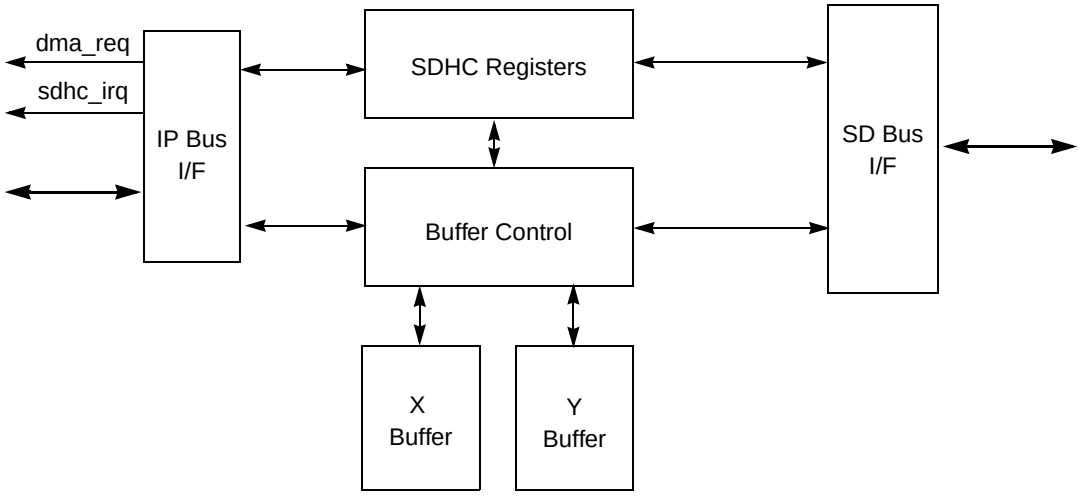


Figure 28-16. SDHC Buffer Scheme

For a host read operation, the SDHC automatically transfers data into the next available buffer. If one of the data buffers is full, the SDHC generates a DMA request. Conversely, for a host write operation, if one of the data buffers is empty, the SDHC generates DMA requests. If some data is available, the SDHC reads the data out of the buffer and writes it to the card through the SD bus interface.

28.4.1.1 Data Buffer Access

The DMA/CPU accesses the SDHC data buffer as a FIFO through the 32-bit Data Buffer Access (SDHC_BUFFER_ACCESS) register. Internally, the SDHC maintains a pointer into the data buffer. Accesses to the SDHC_BUFFER_ACCESS register automatically increase the pointer value. The pointer value is not directly accessible by the software. In cases when the block length of the data transfer is not a multiple of 32 bits, the last access to the SDHC_BUFFER_ACCESS register contains valid data only on 8, 16, or 24 bits. Because the SDHC_BUFFER_ACCESS register only allows 32-bit accesses, put/fetch the data bytes on the correct byte position of the SDHC_BUFFER_ACCESS register. For an 8-bit data access to the FIFO, put/fetch data into SDHC_BUFFER_ACCESS bits 7 through 0. For 16-bit data access, put/fetch data in SDHC_BUFFER_ACCESS bits 15 through 0. For a 24-bit data, put/fetch data into SDHC_BUFFER_ACCESS bits 23 through 0.

When data is written to the card, a 32-bit data word in the data buffer is shifted out from the LSB byte to the MSB byte. When data is read from the card, the data is shifted to the data buffer from LSB byte to MSB byte. See Figure 28-17 for the data swap between system IP bus and SDHC data buffer.

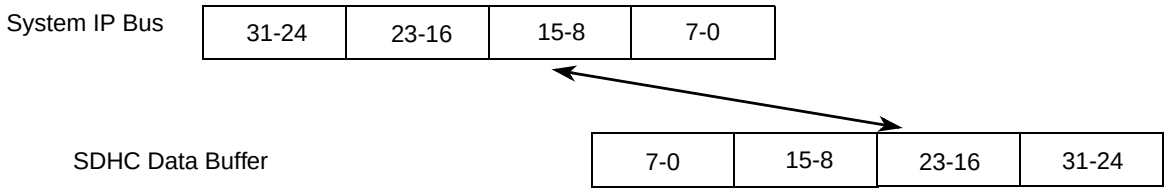


Figure 28-17. Data Swap Between System IP Bus and SDHC Data Buffer

28.4.1.2 Write Operation Sequence

Two methods are available to move data into the SDHC data buffer when writing data to the card. One is by using DMA through the SDHC DMA request signal, and the other is by using the CPU through the STATUS[BUF_WR_RDY] bit (interrupt or polling).

The SDHC automatically asserts a DMA request when the data buffer is empty and it is ready to receive new data. At the same time, SDHC sets the STATUS[BUF_WR_RDY] bit. The buffer write ready interrupt is generated if enabled by software.

The data buffer accumulates the data written until it fills. The SDHC does not start writing to the card until a data buffer size is full. Then, the SDHC starts a transmission when the SD bus is ready for a new transfer. When the other data buffer is empty and more data can be transferred, SDHC asserts a new DMA request and sets the STATUS[BUF_WR_RDY] bit. See Section 28.4.2.1, “DMA Request.”

28.4.1.3 Read Operation Sequence

Two methods are available to fetch data from the buffer when data is read from the card. One is by using DMA through the SDHC DMA request signal, and the other is by using CPU through the BUF_READ_RDY (STATUS[7]) bit (interrupt or polling).

The SDHC asserts a DMA request when the data buffer is full and it is ready for DMA/CPU to fetch data out of the data buffer. At the same time, SDHC sets the BUF_READ_RDY (STATUS[7]) bit. The buffer read ready interrupt is generated if enabled by software.

SDHC only starts receiving data when either of the two data buffers is empty. The buffer accumulates data read from the card until it fills. SDHC asserts a DMA request when either one of the data buffers is full. For multiple block data transfers, while the DMA/CPU moves data by reading the DBA register, SDHC receives data into the other buffer if empty and the SD bus is ready. If the DMA/CPU does not keep up with reading data out of the buffers, SDHC stops the SD_CLK at the block gap to avoid an overflow situation.

28.4.1.4 Data Buffer Size

Buffer sizes must be known during data transfers. In SDHC, both data buffers are 64 bytes in size. However, each data buffer is divided into four 16-byte containers that correspond to the four data lines of the SD bus. Therefore, the data buffer size is 64 bytes in 4-bit SD mode and 16 bytes in 1-bit SD mode.

During multi-block data transfer, block length should be an integer multiple of the buffer size. The buffer is ready to be read by CPU/DMA when either of the buffers is full (STATUS[YBUF_FULL] or STATUS[XBUF_FULL] is set, and STATUS[BUF_READ_READY] is set). The buffer would be ready to write by CPU/DMA (STATUS[YBUF_EMPTY] or STATUS[XBUF_EMPTY] is set, and STATUS[BUF_WR_RDY] is set) when the full buffer of data are fetched out of the buffer. The buffer ready status bit and DMA request are set accordingly.

For single block data transfers when the block length is smaller than the buffer size or when the block length is not an integer multiple times that of the buffer size, the data size may need to be written to the buffer or to be fetched out of the buffer in smaller records. In this case, the buffer would be full (SDHC set STATUS[YBUF_FULL] or STATUS[XBUF_FULL]) when these data are written to the buffer. The buffer

would be empty (the SDHC sets STATUS[YBUF_EMPTY] or STATUS[XBUF_EMPTY]) when these buffer of data are fetched out of the buffer. The STATUS[BUF_READ_RDY] status bit and DMA request would be set accordingly. From the software point of view, the buffer size becomes variable and equal to the real data size that needs to be transferred. This eases the software programming of the SDHC, because inserting dummy data to fill the buffer is not necessary.

28.4.1.5 Dividing Large Data Transfer

This SDIO command CMD53 definition limits the maximum size of data transfers according to this formula.

$$\text{Max Transfer Size} = \text{Block Size} \times \text{Block Count} \qquad \textit{Eqn. 28-1}$$

The block size can be a multiple of the size of the data buffer. However, it is recommended that the block size be equal to the data buffer size. This allows the SDHC to stop the SD_CLK during block gaps should an overflow or underrun condition occur. Stopping the SD_CLK while the data lines are active may cause data corruption on some cards. If an application or card driver needs to transfer larger sets of data, the host driver should divide the data set into multiple blocks.

The length of a multiple block transfer needs to be in block size units. If the total data length cannot be divided evenly to a multiple of the block size, there are two ways to transfer the data depending on function and card design. Option one is for the card driver to split the transaction into a block transfer to send most of the data and a byte transfer to send the remaining data. Option two is to add filler data in the last block to complete the block size. For option two, the card must manage to remove the filler data.

See Figure 28-18 for an example showing dividing a large data set. The 544-byte WLAN frame is divided into eight 64-byte blocks plus the block. Eight 64-byte blocks are sent in block transfer mode and the remaining 32 bytes are sent in byte transfer mode.

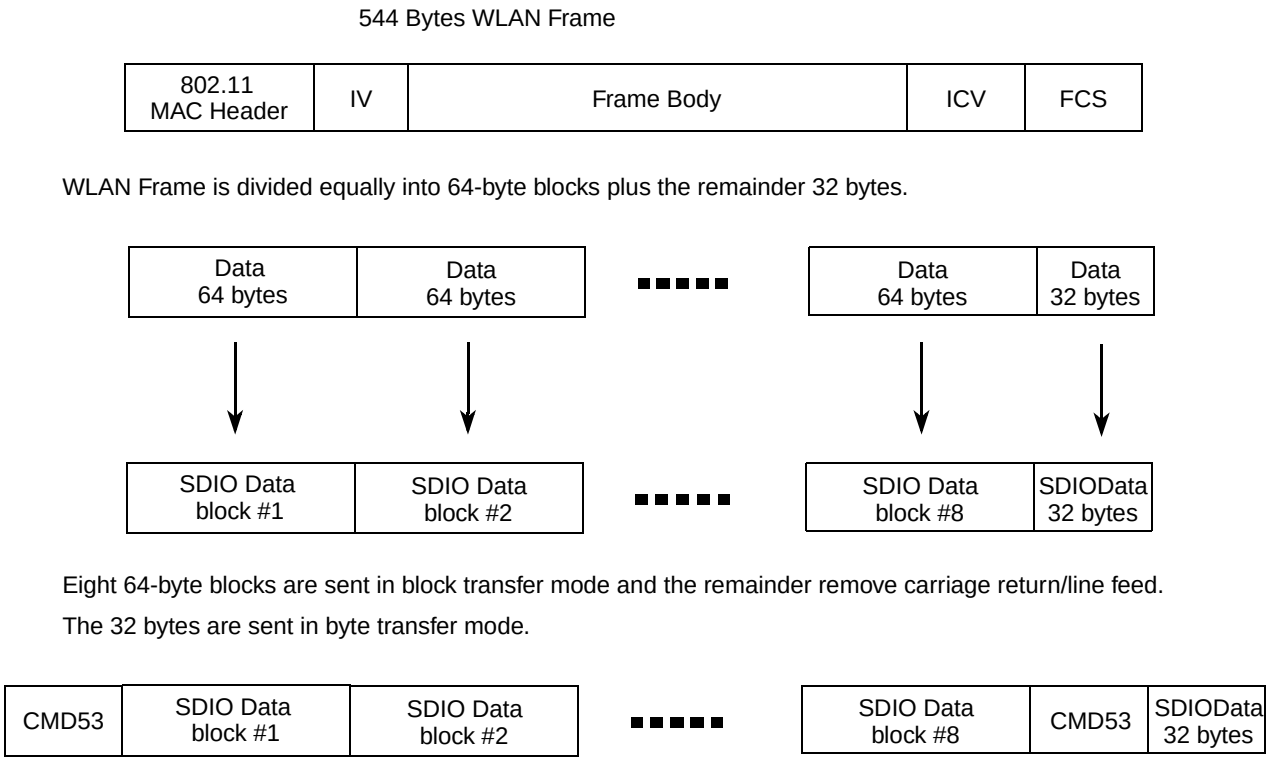


Figure 28-18. Example for Dividing Large Data Transfer

28.4.2 DMA Interface

The DMA interface block controls all data routing between the external data bus (DMA access), internal SDHC 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 SDHC module and the application. See Figure 28-19 for illustration of the DMA interface block.

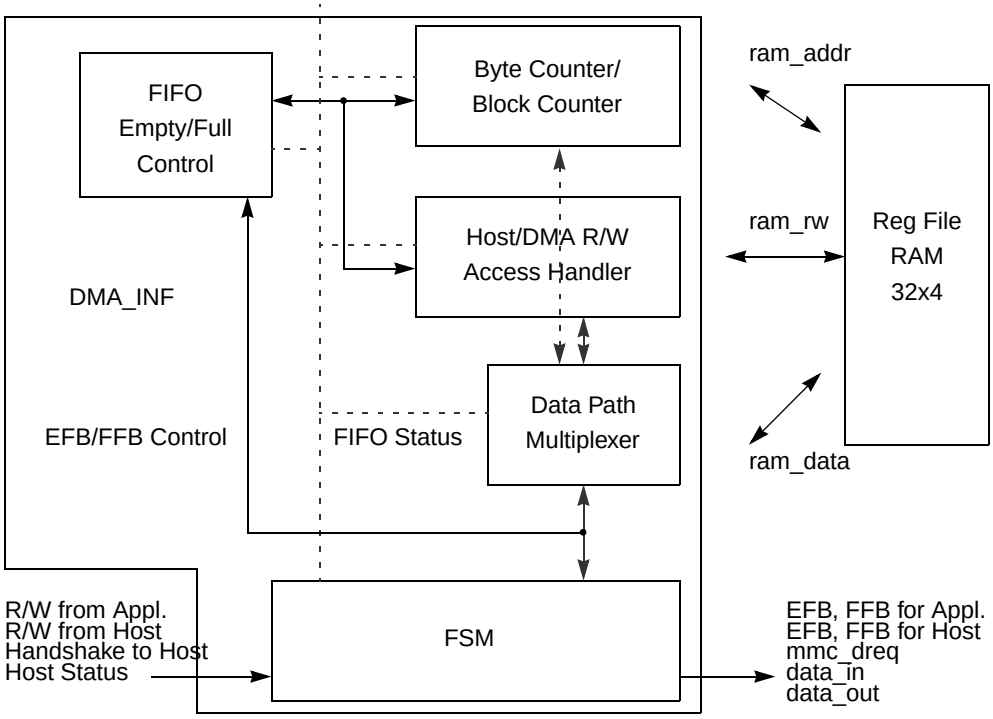


Figure 28-19. DMA Interface Block

In addition, this block also manages the burst request to the external DMA controller, internal register write-error detection, read wait handling of SDIO, and all IP related output responses.

28.4.2.1 DMA Request

If a transfer is in progress, the SDHC generates DMA requests according to the FIFO status. During read operations, the SDHC generates DMA requests if one of the data FIFOs is full. During write operations, the SDHC generates DMA requests if one of the data FIFOs is empty.

To avoid buffer under-run conditions during a write operation, the MMC_SD_CLK stops automatically when both buffers are empty. After the DMA or CPU completes writing data into one of the buffers, MMC_SD_CLK resumes automatically to continue the data transfer.

Similarly, to avoid buffer over-flow during read operations, the MMC_SD_CLK stops automatically when both buffers are full. After the DMA or CPU moves the data out of the buffer, the MMC_SD_CLK resumes automatically to continue the data transfer.

28.4.3 Memory Controller

This controller provides the SDIO-IRQ and read/wait service handling, card detection, command response handling, all SDHC interrupt managing, and it contains the register table. See Figure 28-20 for the block diagram for the memory controller.

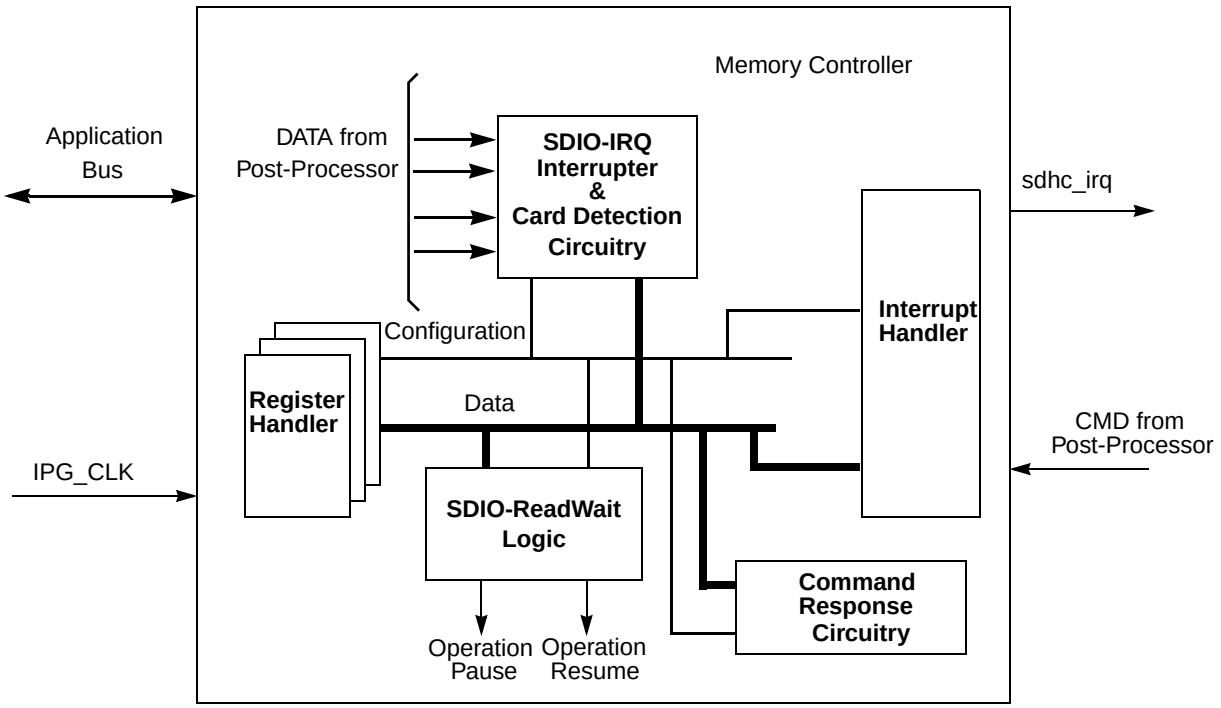


Figure 28-20. Memory Controller Block Diagram

A summary of events that take place when a SDIO card generates an interrupt is detailed in this section. When an SDIO card generates an interrupt request, it sets its interrupt pending bit in the CSR register and asserts the interrupt line, which is shared with DAT1 line in 4-bit mode. The SDHC detects and steers the card's interrupt to the selected IRQ line of the interrupt controller.

28.4.4 SDIO Card Interrupt

28.4.4.1 Interrupts in 1-Bit Mode

In this case, the DAT1 pin is dedicated to providing the interrupt function. Pulling the DAT1 low asserts an interrupt until the host clears the interrupt.

28.4.4.2 Interrupt in 4-Bit Mode

Because the interrupt and data line 1 share pin DAT1 in 4-bit mode, an interrupt is only sent by the card and recognized by the host during a specific time. This is known as the interrupt period. The SDHC samples only the level on DAT1 during the interrupt period. At all other times, the host interrupt controller ignores the level on DAT1. The definition of the interrupt period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the interrupt period becomes active two clock cycles after the completion of a data packet. This interrupt period lasts until after the card receives the end bit of the next command with a data block transfer associated with it.

For multiple block data transfers in 4-bit mode, the interrupt period can be active for only a limited period of time, due to the limited period of data line availability between multiple blocks of data. This requires a more strict definition of the interrupt period. For this case, the interrupt period is limited to two MMC_SD_CLK clock cycles. This begins two clocks after the end bit of the previous data block. During this 2-clock cycle interrupt period, if an interrupt is pending, the DAT1 line is held low for one clock cycle with the last clock cycle pulling DAT1 high. On completion of the interrupt period, the card releases the DAT1 line into the high impedance state.

When in 4-bit mode, the SDHC differentiates a data start bit and the interrupt period by checking all four data lines are low for the start of new data. In the case of an interrupt, only the DAT1 should have gone low. After the last data block is sent, the interrupt period starts as normal after the end of this data block, but it ends after the next command with a data block commences, instead of lasting two cycles.

Refer to the SDIO Card Specification for further information about the SDIO card interrupt.

28.4.4.3 Card Interrupt Handling

When the SDIO_IRQ_EN bit in the SDHC Interrupt Control register (SDHC_INT_CNTR) is set to 0, the host controller clears the interrupt request to the system interrupt controller. The SDIO Interrupt detection is stopped when this bit is cleared and restarted when this bit is set to 1. The host driver should clear the SDIO interrupt enable bit before servicing the SDIO interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO Status bit is cleared by resetting the SDIO interrupt. Writing to this bit has effect in 1-bit mode, as the host controller detects the SDIO interrupt with or without SD clock (to support wakeup). In 4-bit mode, the interrupt signal is sampled during the interrupt period; therefore, some sample delays exist between the interrupt signal from the SDIO card and the interrupt to the host system interrupt controller. When the SDIO status has been set and the host driver needs to start this interrupt service, the SDIO bit in the interrupt control register is set to 0 to clear the SDIO interrupt status latched in the SDHC and to stop driving the interrupt signal to the system interrupt controller. The host driver must issue a CMD52 to clear the interrupts at the card. After completion of the card interrupt service, the SDIO interrupt enable bit is set to 1 and SDHC starts sampling the interrupt signal again. See Figure 28-21 (a) for illustration of the SDIO card interrupt scheme and Figure 28-21 (b) for the sequences of software and hardware events taking place during card interrupt handling procedure

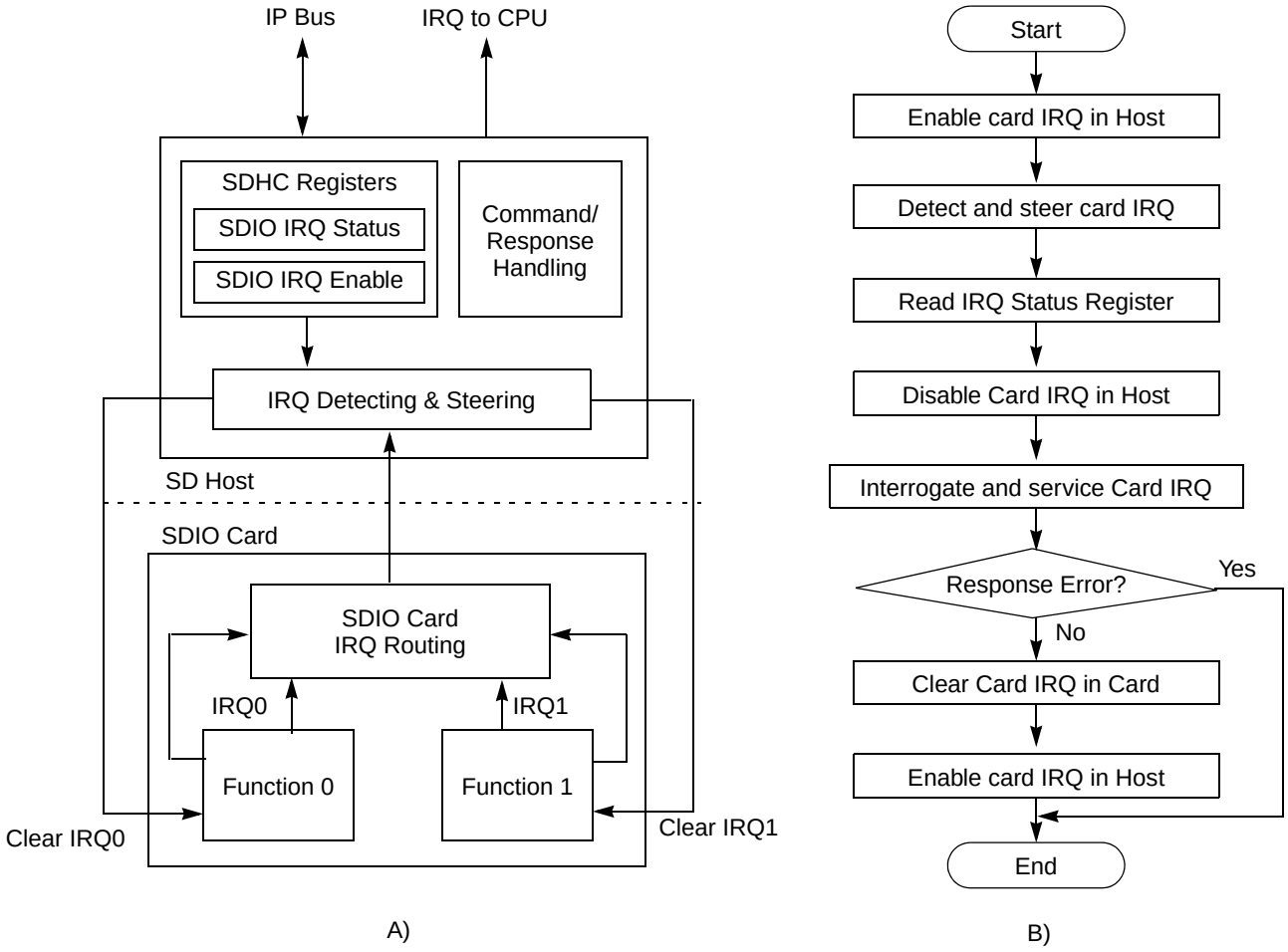


Figure 28-21. A) Card Interrupt Scheme; B) Card Interrupt Detection and Handling Procedure

28.4.5 Card Insertion and Removal Detection

SDHC uses the DAT3 pin to detect card insertion or removal. To use this feature of the SDHC, chip level integration needs to pull-down this pad as a default state. When no card exists on the MMC/SD bus, DAT3 defaults to a low voltage level. When any card is inserted to or removed from the socket, SDHC detects the logic value changes on the DAT3 pin and generates an interrupt.

Because the mechanism is based on the value of the DAT3 line, only single-card systems can benefit from card detection. To avoid conflicts of card insertion/removal detection and the data value changes on DAT3 due to data transfer, disable the card insertion interrupt when a card is detected in the socket and enabled when the card is removed from the socket. The card removal interrupt can only be enabled when no bus activity occurs on DAT3.

To avoid false status bit generation during data transfer, the card insertion/removal is masked by corresponding interrupt enable bits in the SDHC_INT_CNTR register.

Above all, there are three interrupt sources: card removal interrupt, card insertion interrupt, and SDIO card interrupt. All interrupt sources are ORed between the peripheral and the interrupt controller.

NOTE

Send a command (CMD42 for SDMem or CMD52 for SDIO) to the card to disable the card internal pull-up resistor after card detection and identification. Because the SD protocol requires the DAT line must be pulled up for data transfer, disable the host side of the DAT3 pull-down feature and configure it as pull-up. If the card internal pull-up resistor is disabled during this, the card removal interrupt cannot be detected through DAT3.

28.4.6 Power Management

When there is no operation between SDHC and the card through SD bus, disable the ipg_clk and SDHC_CLK in chip level clock control module to save power. When you need to use SDHC to communicate with the card, enable the clock.

28.4.7 System Clock Controller

There is one clock divider and one clock prescaler in SDHC to divide the high frequency input clock SDHC_CLK to a lower frequency clock, which most of the SDHC logic can use. See Figure 28-22 for details about clocks used in SDHC. The input clock first goes through a 4-bit divider and then a 12-bit prescaler to generate a clock named CLK_20M. This clock is used internally by SDHC and generates the MMC_SD_CLK. The MMC_SD_CLK to the card has the same clock frequency as CLK_20M. CLK_20M is derived from the CLK_DIV by using the 12-bit prescaler. The CLK_DIV is derived from the input clock SDHC_CLK by using the 4-bit divider. SDHC clock rate register controls the divide rate for both the divider and the prescaler. Refer to Section 28.3.2.3, “SDHC Clock Rate (SDHC_CLK_RATE) Register,” for the clock rate register information.

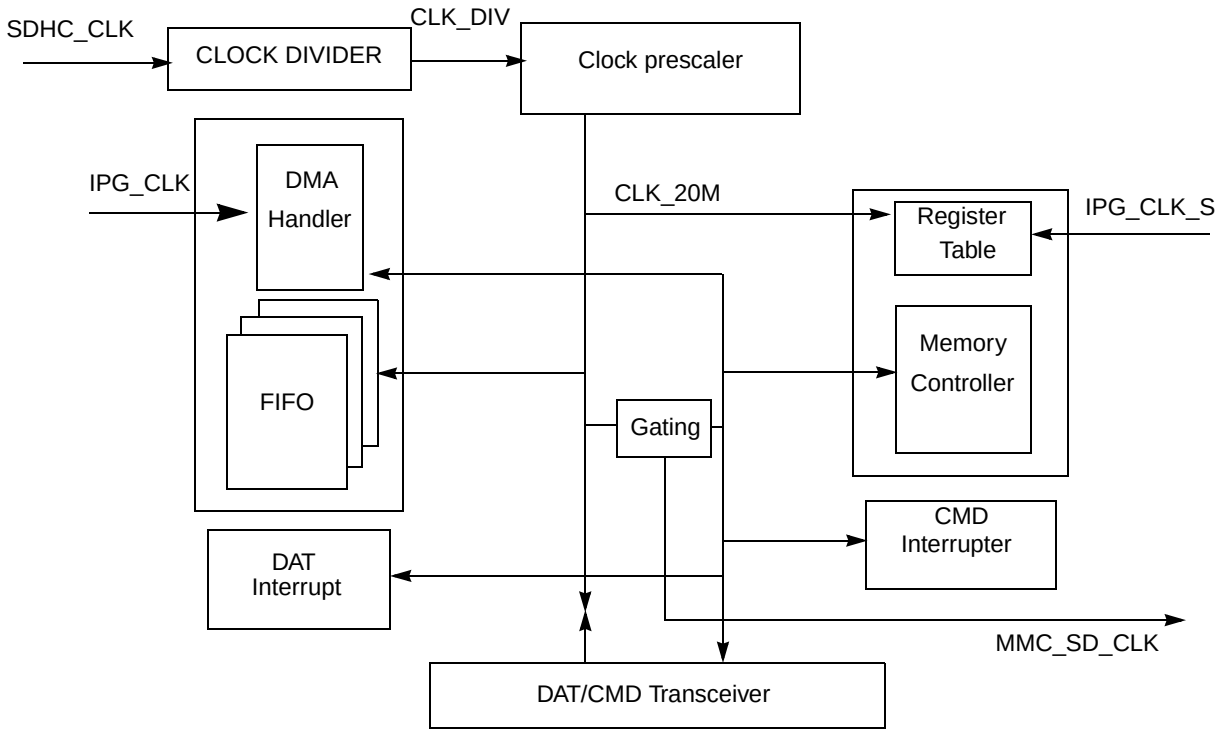


Figure 28-22. Clock Used in SDHC

To get the maximum power-saving during the operation, the SDHC bus clock pauses and resumes according to the SDHC status. For example, when FIFO is full during the card read operation, the bus clock is stopped if no further data is written to FIFO by card. It is resumed when user (DMA) clears FIFO empty status; similarly, there are other conditions where SDHC stops the clock to save power.

The controller controls the rate of the host main clock and checks whether it is on or off. The clock is turned off by setting the SDHC_STR_STP_CLK[STOP_CLK] bit and is turned on by setting the SDHC_STR_STP_CLK[START_CLK] bit. To change the clock rate, software needs to write a new value in the SDHC_CLK_RATE register.

28.5 Initialization Information

The host controls all communication between system and cards. Also, the host sends commands of two types: broadcast and addressed (point-to-point) commands.

Broadcast commands are intended for all cards, such as: Go_Idle_State, Send_Op_Cond, All_send_CID, and Set_relative_Addr. In broadcast mode, all cards are in the open-drain mode to avoid bus contention. If the socket supports only one card, the broadcast command is similar as the point-to-point command.

After the broadcast command Set_relative_Addr is issued, all cards enter standby mode. Addressed type commands are used from this point on. In this mode, the CMD/DAT I/O returns to push-pull mode to have the driving capability for maximum frequency operation.

As mentioned in the above section, MMC and SD are similar products. Other than the 4x bandwidth, they are programmed similarly. The following example shows how to initialize and perform content access and content protection on the cards.

To improve the readability, use a program-like function for the example.

28.5.1 MMC_SD_CLK Control

The SDHC_STR_STP_CLK register controls the MMC_SD_CLK clock to the card. The clock should be supplied to the card for:

- Submitting command to card and receiving response
- Transferring data between SDHC and the card
- Detecting an interrupt from a SD card in 4-bit

The steps below show how to start the MMC_SD_CLK to card:

1. Write 0x2 to the SDHC_STR_STP_CLK register.
2. Poll the SDHC_STATUS[CARD_BUS_CLK_RUN] bit, and wait until the clock starts.

The steps below show how to stop MMC_SD_CLK to card (it is not recommended to stop clock by software):

1. Write 0x1 to the SDHC_STR_STP_CLK register.
2. Poll the SDHC_STATUS[CARD_BUS_CLK_RUN] bit, and wait until the clock is stopped.

28.5.2 Command Submit – Response Receive Basic Operation

Below is the program flow to submit a command to the card(s). Targeted command is <command_no>. Corresponding argument is <arg_no>. The command configuration required is <cmd_dat_cont>. The interrupt control used in software is <int_Control_value>.

The steps below show how to submit a command to the card:

1. Start MMC_SD_CLK if it is stopped.
2. Enable the END_CMD_RESP interrupt by writing 0 to SDHC_INT_CNTR[END_CMD_RESP].
3. Set command number to the SDHC_CMD register.
4. Set the command argument to the SDHC_ARG register.
5. Set the appropriate value to the SDHC_CMD_DAT_CONT register.
6. Wait for the command response end interrupt and check for the response CRC/time-out status.
7. Read the response FIFO to check the response. Read three or eight times from the response FIFO access register, depending upon whether the response is 48-bit or 136-bit.

This following is a function defining command submission. This function is used in the examples in the following sub-section:

```
send_cmd_wait_resp(command_no, arg, cmd_dat_cont, int_cntr_value)
{
write_reg(COMMAND, <command_no>); // 1. configure the CMD
write_reg(SDHC_ARG, <arg_no>); // 2. configure the command argument
```

```
write_reg(SDHC_CMD_DAT_CONT, <cmd_dat_cont>); // 3. configure the command data
control register, writing to this register triggers SDHC send command to the
card.
while(irq_status); // 4. Wait interrupt (End Command Response)
Write_reg(SDHC_INT_CNTR, <int_cntr_value>); // 5.  irq request from SDHC
read_reg(STATUS); // 6. Check whether the interrupt is an End_CMD_RES or a
response time out or a CRC error.
read_reg(SDHC_RES_FIFO); // 7. read the response FIFO to determine if the command
has a response
}
```

28.5.3 Card Identification Mode

When a card is inserted to the socket or the card was reset by the host, the host needs to validate the operation voltage range, identify the cards, and request the cards to publish the relative card address (RCA) or to set the RCA for the MMC cards. All data communication in the card identification mode uses the command line (CMD) only.

28.5.3.1 Card Detect

See Figure 28-23 for a flow diagram showing the card detection using the host controller.

- Write 1 to SDHC_INT_CNTR[CARD_INSERTION_EN] to enable card detection interrupt
- Write 0 to SDHC_INT_CNTR[CARD_REMOVAL_EN] to disable card detection interrupt

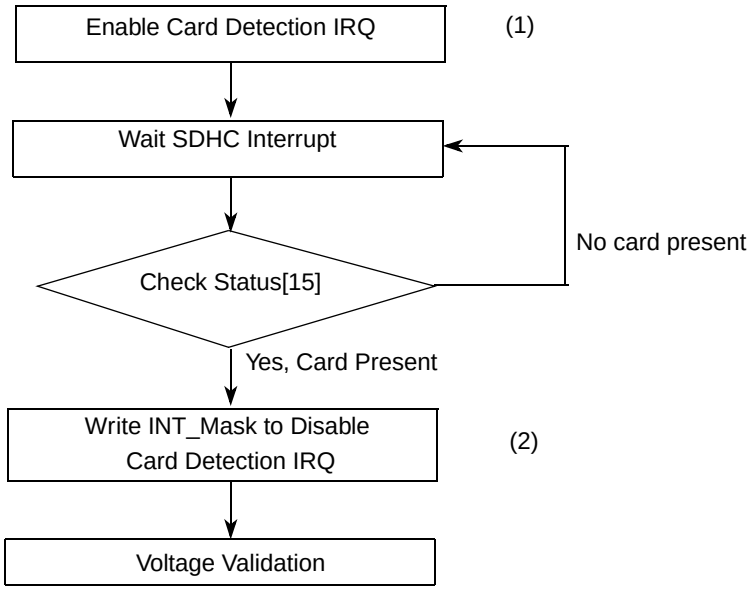


Figure 28-23. Flow Diagram for Card Detection

28.5.3.2 Reset

There are three types of reset:

- Hardware reset (Card and Host) driven by POR (power on reset).

- Software reset (Host Only) proceeds by the write operation on the SDHC_STR_STP_CLK register. Follow the recommended sequence as specified in Section 28.3.2.1, “SDHC Clock Control (SDHC_STR_STP_CLK) Register.” The reset can reset all the SDHC registers, but does not reset the card. The card reset is through CMD0. After you apply software reset to SDHC, it should also use CMD0 to reset the card in case the card is in an unknown state. Write 0x2 to the SDHC_STR_STP_CLK register and poll status[8] to wait until the clock is on. It is highly recommended to start clock here and leave the clock unchanged afterwards.
- Card is reset (Card Only). The command, Go_Idle_State, CMD0 is the software reset command for the MMC and SD memory card. This sets each card into idle state regardless of the current card state. When used as a SD I/O Card, CMD52 writes IO reset in CCCR. The cards are initialized with a default relative card address (RCA = 0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. See Figure 28-24 for the software flow to reset both SDHC and the card.

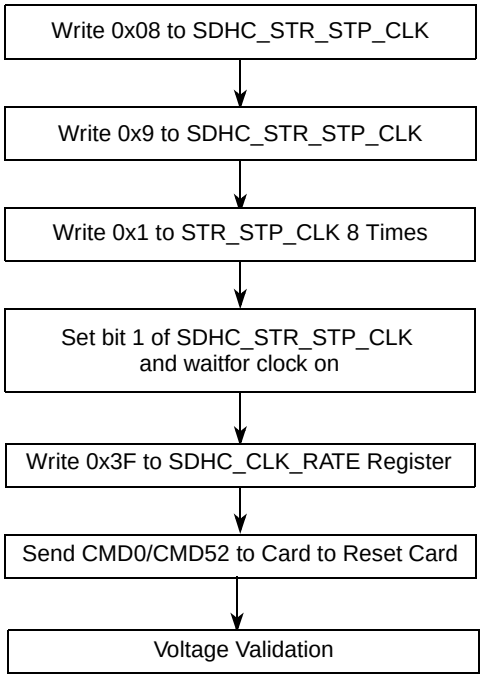


Figure 28-24. Flow Chart for Reset of SDHC and SD I/O Card

```
software_reset()
{
write_reg(SDHC_STR_STP_CLK, 0x8);
write_reg(SDHC_STR_STP_CLK, 0x9);// 1. reset the SDHC host;
write_reg(SDHC_STR_STP_CLK, 0x1);
write_reg(SDHC_STR_STP_CLK, 0x1);
write_reg(SDHC_STR_STP_CLK, 0x1);
write_reg(SDHC_STR_STP_CLK, 0x1);
```

```
write_reg(SDHC_STR_STP_CLK, 0x1);
write_reg(SDHC_STR_STP_CLK, 0x1);
write_reg(SDHC_STR_STP_CLK, 0x1);
write_reg(SDHC_STR_STP_CLK, 0x1);// 2. write 0x1 to SDHC_STR_STP_CLK 8 times;
write_reg(SDHC_STR_STP_CLK, 0x2); // 3. write 0x2 to SDHC_STR_STP_CLK to start clock;
while (!STATUS[8]); // 4. wait clock on
write_reg(SDHC_CLK_RATE, 0x3F);  // 5. Set the lowest clock for initialization
write_reg(SDHC_READ_TO, 0x2DB4); // 6. set READ timeout register
send_cmd_wait_resp(CMD_GO_IDLE_STATE, 0x0,0x80, 0x40); //7. reset the card with CMD0
}
```

28.5.3.3 Voltage Validation

All cards are able to establish communication with the host using any operation voltage in the allowed voltage range specified in the standard. However, the supported minimum and maximum voltages are defined in the operation conditions register (OCR) and may not cover the whole range. Cards that store the CID and CSD data in the preload memory can only communicate this information under data transfer V_{DD} conditions. If the host and card have non-compatible voltage ranges, the card cannot complete the identification cycle nor send CSD data.

Therefore, a special command Send_Op_Cont (CMD1 for MMC), a SD_Send_Op_Cont (CMD41 for SD Memory), and a IO_Send_Op_Cont (CMD5 for SD I/O) provide a mechanism to identify and reject cards which do not match the voltage range desired by the host. The host accomplishes this by sending the required voltage window as the operand of this command. Cards which cannot perform data transfer in the specified range must disable themselves from further bus operations and go into the inactive state. By omitting voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into inactive state. This query should be used if the host can select a common voltage range or if a notification to the application of non-usable cards in the stack is desired.

The following steps show how to perform voltage validation when a card is inserted:

```
voltage_validation(voltage_range_arguement)
{
send_cmd_wait_resp(IO_SEND_OP_COND, 0x0, 0x04, 0x40); // CMD5, send SDIO operation
voltage, command argument is zero
if(End Command Response true & No. of IO functions> 0)// it is SDIO and have IO function
{IORDY = 0;
while(!(IORDY in I/O ORC response)) {// set voltage range for each IO
send_cmd_wait_resp(IO_SEND_OP_COND, voltage_range_arguement, 0x04, 0x40);}
if(Memory Present flag true)
Card = combo; // that is, SDIO + SD Memory, need to set operation voltage to memory
portion as well
send_cmd_wait_resp(APP_CMD, 0x0, 0x01, 0x40);// CMD55, Application Command follows
send_cmd_wait_resp(SD_APP_OP_COND, voltage_range_argument, 0x01, 0x40);//ACMD41
else
Card = sdio;

// if No response to CMD5 IO_SEND_OP_COND or No. of IO Function is zero in response
else// the card should be SD or MMC
{send_cmd_wait_resp(APP_CMD, 0x0, 0x01, 0x40);// CMD55, Application Command follows
```

```
if(End Command Response true and no response timeout)
{send_cmd_wait_resp(SD_APP_OP_COND, voltage_range_arguement, 0x01, 0x40); // ACMD41, SD
card found
Card = sd;
}
else // the card have no response to APP_CMD, it is not SD card
{send_cmd_wait_resp(SEND_OP_COND, voltage_range_arguement, 0x01, 0x40); //CMD1, MMC card
found
if(End Command Response true and no response timeout)
{Card = mmc;}
else{ Card = No card or failed contact;}
                                                                                }
}
```

28.5.3.4 Card Registry

Card registry on MMC and SD cards is different.

For SD card operation, the identification process starts at clock rate Fod, initialization clock frequency defined by the card spec, (below 400 kHz for most of the card) as defined by the card spec. After the bus is activated, the host requests the card to send their valid operation conditions. The response to ACMD41 is the operation condition register of the card. The same command is sent to all of the new cards in the system. Incompatible cards are put into inactive state. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card identification (CID) number. Cards currently unidentified (in ready state) send their CID number as the response. After the CID is sent by the card, the card goes into the identification state.

The host then issues Send_Relative_Addr (CMD3), requesting the card to publish a new relative card address (RCA) shorter than CID. This addresses the card for future data transfer operations. After the RCA is received, the card state changes to the stand-by state. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system.

For MMC operation, the host starts the card identification process in open-drain mode with the identification clock rate Fod. Open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is activated, the host requests the cards to send their valid operation conditions (CMD1). The response to CMD1 is a wired operation on the condition restrictions of all cards in the system. Incompatible cards are sent into inactive state. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards (those in ready state) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bitstream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any of the bit periods, stop sending their CID immediately and must wait for the next identification cycle. Because CID is unique for each card, only one card can successfully send its full CID to the host. This card then goes into identification state. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to this card a relative card address (RCA). After the RCA is received, the card state changes to stand-by state, and the card does not react to further identification cycles. Its output switches from open-drain to push-pull. The host repeats the process, CMD2 and CMD3, until the host receives time-out condition to recognize completion of the identification process.

```
card_registry()
{
while (ResponseTO from STATUS){
if(card==combo or sdio)
{
send_cmd_wait_resp(SET_RELATIVE_ADDR, 0x00, 0x01, 0x40);   //card publish the RCA in
response
rca = SDIO_RCA = address from response FIFO;
}
else if(card==sd)
{
send_cmd_wait_resp(ALL_SEND_CID, 0x00, 0x02, 0x40);
send_cmd_wait_resp(SET_RELATIVE_ADDR, 0x00, 0x01, 0x40); //card publish the RCA in
response
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_arguement, 0x01, 0x40);
}
else
exit due to card not identified;
}
send_cmd_wait_resp(SELECT_CARD, RCA_arguement, 0x41, 0x40);
}
```

28.5.4 Card Access

28.5.4.1 Block Access — Block Write & Block Read

28.5.4.1.1 Block Write

During block write, (CMD24–27) one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. A card supporting block write can always accept a block of data defined by WRITE_BL_LEN. If the CRC fails, the card indicates the failure on the DAT line (see below); the transferred data is discarded and not written, and all further transmitted blocks (in multiple block write mode) are ignored.

If the host uses partial blocks with accumulated length not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card shall detect the block misalignment error and abort programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer, waits in the receive-data-state for a stop command. The write operation is also aborted if the host tries to write over a write protected area. In this case, however, the card sets the WP_VIOLATION bit.

Programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, this unchangeable part must match the corresponding part of the receive buffer. If this match fails, the card reports an error and does not change any register contents. Some cards may require unpredictable times to write a block of data. After receiving a block of data and completing the CRC check, the card begins

writing and holds the DAT line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) at any time, and the card responds with its status. The status bit READY_FOR_DATA indicates the card can accept new data or the write process remains in progress. The host may deselect the card by issuing CMD7 (to select a different card) which displaces the card into the disconnect state and releases the DAT line without interrupting the write operation. When re-selecting the card, it reactivates the busy indication by pulling DAT to low if programming remains in progress and the write buffer is unavailable.

The software flow to write to card with DMA enable is:

1. Start MMC_SD_CLK if it is stopped.
2. Check the card status, wait until card is ready for data.
3. For SD/MMC, set the card block length, using SET_BLOCKLEN (CMD16).
4. Set the SDHC block length register to be same as block length set to the card in Step 3. For SDIO, if the CMD53 is in byte mode, the SDHC block length register should be set according to bytes count in CMD53; if the CMD53 is in block mode, the SDHC block length register should be set according to the block size in CCCR registers.
5. Set the SDHC Number of Blocks (SDHC_NOB) register. NOB is 1 for single block write or CMD53 in byte mode for SDIO.
6. Disable the buffer ready interrupt, configure the DMA setting and enable the SDHC DMA channel:
   - Write 1 to bit[3] of INT_MASK register in SDHC.
   - Set DMA destination to be SDHC_Buffer Access register.
   - Set DMA destination port size to be 32-bit.
   - Set DMA Burst length to be 16 bytes in 1-bit mode or 64 bytes in 4-bit mode.
   - Set DMA transfer count to be number of bytes which is a multiple of the Block_length (NOB × blk_len = total number of bytes).
7. Check the card status and wait until the card is ready for data.
8. Set the SDHC_CMD register to any of the following:
   - CMD24(WRITE_BLOCK), or
   - CMD25(WRITE_MULTIPLE_BLOCK), or
   - CMD53 in byte mode or block mode
9. Set the SDHC Command Argument (SDHC_ARG) register.
10. Set the SDHC Command Data Control (SDHC_CMD_DAT_CONT) register.
11. Wait for end command response and check if there any CRC error or timeout error.
12. Wait for DMA done.
13. Check for Write_OP_DONE and check status bit to see if write CRC error occurred.
14. Send STOP_TRANSMISSION command to card if the write command is WRITE_MULTIPLE_BLOCK (CMD25).

If the write operation is without DMA, the system needs to write data to the buffer through buffer write ready interrupt or by polling the buffer write ready status bit (STATUS[BUF_WR_RDY]). For high performance, data transfer using DMA is preferred.

28.5.4.1.2 Block Read

For block reads, the basic unit of data transfer is a block whose maximum size is defined in the CSD (READ_BL_LEN). If READ_BL_PARTIAL is set, smaller blocks whose starting and ending address are entirely contained within one physical block (as defined by READ_BL_LEN) may also be transmitted. A CRC is appended to the end of each block, ensuring data transfer integrity. CMD17 (READ_SINGLE_BLOCK) initiates a block read. After completing the transfer, the card returns to the transfer state. CMD18 (READ_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. Blocks are continuously transferred until a stop command is issued. If the host uses partial blocks with accumulated length not block aligned and block misalignment is not allowed, the card detects a block misalignment at the beginning of the first mis-aligned block, sets the ADDRESS_ERROR error bit in the status register, aborts transmission, and waits in the data state for a STOP command.

The software flow to write to card with DMA enable is:

1. Start MMC_SD_CLK if it is stopped.
2. Check the card status and wait until the card is ready for data.
3. For SD/MMC, set the card block length, using SET_BLOCKLEN (CMD16).
4. Set the SDHC block length register to be same as block length set to the card in Step 3.

   For SDIO, if the CMD53 is in byte mode, the SDHC block length register should be set according to bytes count in CMD53; if the CMD53 is in block mode, the SDHC block length (SDHC_BLK_LEN) register should be set according to the block size in CCCR registers.
5. Set the SDHC Number of Blocks (SDHC_NOB) register to 1 for single block write or CMD53 in byte mode for SDIO.
6. Disable the buffer ready interrupt, configure the DMA setting, and enable the SDHC dma channel:
   - Write 0 to bit[4] of the SDHC_INT_CNTR register in the SDHC to disable the buffer read ready interrupt.
   - Set DMA source to be SDHC_Buffer Access register.
   - Set DMA source port size to be 32-bit.
   - Set DMA Burst length to be 16 bytes in 1-bit mode or 64 bytes in 4-bit mode.
   - Set DMA transfer count to be number of bytes which is a number of blocks multiple of the Block_length (NOB × blk_len).
7. Check the card status and wait until the card is ready for data.
8. Set the SDHC_CMD register to be CMD17(READ_SINGLE_BLOCK) or CMD18 (READ_MULTIPLE_BLOCK) or CMD53 in byte mode or block mode.
9. Set the SDHC Command Argument (SDHC_ARG) egister.
10. Set the SDHC Command Data Control (SDHC_CMDDAT_CONT) register.
11. Wait for END_CMD_RESP interrupt and check response FIFO, check CRC error and timeout error.
12. Wait for DMA done.
13. Check for READ_OP_DONE and check status bit to see if read CRC error occurred.
14. Send STOP_TRANSMISSION command to card if the read command is READ_MULTIPLE_BLOCK (CMD18).

If the read transfer operation does not use DMA, the system needs to fetch data out of the data buffer through utilizing the buffer read ready interrupt or by polling the STATUS[BUF_READ_RDY] status bit. For high performance, data transfer using DMA is preferred.

28.5.5 Switch Card Mode

Switch function command (CMD6) switches or expands memory card functions. Two function groups are defined:

- Card access mode: 12.5 MB/s interface speed (default) or 25 MB/s interface speed. (high speed).
- Card command system: Standard command set (default), eCommerce command set, or Vendor Specific Command set.

This is a new feature, introduced in SD Specifications Part 1 PHYSICAL LAYER Specification Version 1.10, so cards compatible with earlier version do not support this feature. Before issuing CMD6 to switch-card mode, the host driver checks SD_SPEC field in SCR register to confirm the command is supported. Card only accepts CMD6 in transfer state. After selected, all functions only return to the default function after a power cycle, CMD6 (Mode 1 operation with Function 0 in each function group) or CMD0. Executing a power cycle or issuing CMD0 causes the card to reset to the idle state and all the functions to switch back to the default function. On responding to CMD6, the card sends R1 response on the CMD line and 512 bits of status on the DAT lines. Therefore, for the host controller, this is like CMD17 for a single block read with block size of 64 bytes. The time-out value of this command is also 100 ms. If CRC error occurs on the status data, the host driver should issue a power cycle. CMD6 function switching period is within eight clocks after the end bit of status data. When CMD6 changes the bus behavior, the host can use the new functions then.

The software flow to enable high speed mode with DMA enabled is (assume SD_SPEC field is verified):

1. Start MMC_SD_CLK if it is stopped.
2. Check the card status and wait until the card is ready for data.
3. Set the card block length to 64 bytes, using SET_BLOCKLEN (CMD16).
4. Set the SDHC Number of Blocks (SDHC_NOB) register to 1.
5. Disable the buffer ready interrupt, configure the DMA setting, and enable the SDHC DMA channel:
   - Write 0 to bit[4] of the SDHC_INT_CNTR register in the SDHC to disable the buffer read ready interrupt.
   - Set DMA source to be SDHC_Buffer Access register.
   - Set DMA source port size to be 32-bit.
   - Set DMA burst length to be 16 bytes in 1-bit mode or 64 bytes in 4-bit mode.
   - Set DMA transfer count to be 64 bytes.
6. Check the card status and wait until the card is ready for data.
7. Set the SDHC_CMD register to be CMD6(SWITCH).
8. Set the SDHC Command Argument (SDHC_ARG) register to 0x00FF_FFF1.
9. Set the SDHC Command Data Control (SDHC_CMD_DAT_CONT) register.

10. Wait for END_CMD_RESP interrupt and check response FIFO, check CRC error and timeout error.
11. Wait for DMA done.
12. Check for READ_OP_DONE and check status bit to see if read CRC error occurred.
13. Check if bit 401 of received 512 bits is 1 to confirm high speed mode is supported.
14. Repeat steps 6 through 12 except in step 8, set CMD Argument as 0x80FF_FFF1.
15. Check if bits 379~376 are 4’b0001 of received 512 bits to confirm high speed mode is enabled.

Change clock to about 50 MHz for high speed mode.

This page is intentionally left blank.

Chapter 29
Software Watchdog Timer (WDT)

29.1 Introduction

The Software Watchdog Timer (WDT) provides a method to recover from conditions caused by improper software operation. For example, software may become lost due to a programming error or an electrical problem. Also, the software may become trapped in a loop with no controlled exit.

The WDT is a down-counter that is periodically reset to a maximum count by writing a special service sequence to the software watchdog service register (SWSRR). If the WDT decrements to 0x0000, a software reset or a machine check processor exception (MCP) is generated. By default, the WDT is enabled at the release of reset. The WDT can be disabled by setting the SWEN bit to 0. The SWEN bit is in the watchdog control register (WDT_SWCRR).

29.1.1 Features

Key features of the WDT include the following:

- 16-bit prescaler and 16-bit down-counter
- Selectable range for timeout period
- Timeout delay of approximately 128 seconds maximum with a 33 MHz system XTAL clock[1]

29.1.2 Modes of Operation

The WDT unit can operate in the following modes:

- WDT enable/disable mode:

  If the WDT is not needed, the user can disable it with software after a system reset. When it is disabled, the watchdog counter and prescaler counter are held in a stopped state.
- WDT output reset/interrupt mode:

  Without periodic software servicing, the WDT times out and issues a reset or a nonmaskable MCP interrupt
- WDT prescaled/non-prescaled clock mode:

  The WDT counter clock can be prescaled by programming the WDT_SWCRR[SWPR] bit that controls the divide-by-65,536 of the WDT prescaler counter.

1. The system XTAL clock can be different from 33 MHz.

29.2 Memory Map/Register Definition

The WDT programmable register map occupies 16 bytes of memory-mapped space. Reading undefined portions of the memory map returns all 0s; writing has no effect.

It is recommended that user software not access undefined or reserved locations in the programmable register map for the WDT or any other module. Some locations in the programmable register map may be designated as undefined or reserved, but in fact, may contain non-user mode registers used for testing or for modifying the operation of the module.

All WDT registers are 16 or 32 bits wide, located on 16-bit address boundaries, and should be accessed as 16-bit or 32-bit quantities. That is, 16-bit wide registers should be addressed using half-word accesses, and 32-bit wide registers should be addressed using word accesses. All addresses used in this chapter are offsets from the WDT base address, as defined in Section 2.2, "Memory Map and Register Definition."

29.2.1 Memory Map

A memory map of the WDT is shown in Table 29-1.

Table 29-1. WDT memory map

Offset from WDT_BASE (0xFF40E_0900)[1]	Register	Access	Reset Value	Section/Page
0x00	Reserved			
0x04	WDT_SWCRR—Software watchdog control register	R/W	0x0000_0004	29.2.2.1/29-800
0x08	WDT_SWCNR—Software watchdog count register	R	0x0000_0000	29.2.2.2/29-801
0x0C–0x0D	Reserved			
0x0E	SWSRR—Software watchdog service register	W	0x0000	29.2.2.3/29-802
0x10–0xFF	Reserved			

[1] Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, "System Configuration and Memory Map (XLBMEN + Mem Map)."

29.2.2 Register Descriptions

29.2.2.1 Software Watchdog Control Register (WDT_SWCRR)

The Software Watchdog Control Register (WDT_SWCRR), shown in Figure 29-1, controls the software watchdog period and configures WDT operation. The WDT_SWCRR can be read at any time but can be only written once after system reset.

Address: Base + 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SWTC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	SWEN	SWRI	SWPR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1[1]	0	0

Figure 29-1. Software Watchdog Control Register (WDT_SWCRR)

1 On reset, SWEN is set to 1. After reset, the WDT is enabled.

Table 29-2. WDT_SWCRR field descriptions

Field	Description
SWTC	Software watchdog time count. The SWTC field contains the modulus that is reloaded into the watchdog counter by the special service sequence. When a new value is loaded into WDT_SWCRR[SWTC], WDT is not updated until the special service sequence is written to the WDT_SWSRR register. If WDT_SWCRR[SWEN] is loaded with 0, the modulus counter does not count. The new value is also used at the next and all subsequent reloads. Reading the WDT_SWCRR register returns the value in the WDT_SWCRR. Reset initializes the WDT_SWCRR[SWTC] field to 0xFFFF. **Note:** The prescaler counter is reset each time the contents of the SWTC field is loaded into the watchdog counter due to executing a special service sequence or by executing a system reset.
SWEN	Software watchdog enable. The WDT_SWCRR[SWEN] bit enables the WDT. This bit is set on reset, and can be cleared by software after a system reset to disable the WDT. When the timer is disabled, the watchdog counter and prescaler counter are held in a stopped state. 0 WDT disabled. 1 WDT enabled.
SWRI	Software watchdog reset/interrupt select. This bit determines whether a WDT time out causes a hardware reset or machine check interrupt to the core. 0 WDT causes an MCP interrupt to the core. 1 WDT causes a hardware reset.
SWPR	Software watchdog counter prescale bit. Controls the divide-by-65,536 WDT counter prescaler. 0 The WDT counter is not prescaled. 1 The WDT counter clock is prescaled.

29.2.2.2 Software Watchdog Count Register (WDT_SWCNR)

The Software Watchdog Count Register (WDT_SWCNR), shown in Figure 29-2, provides visibility to the watchdog counter value. WDT_SWCNR is a read-only register. Writing to the WDT_SWCNR register has no effect and terminates without transfer error exception.

Address: Base + 0xHHHH Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SWCN															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 29-2. Software Watchdog Count Register (WDT_SWCNR)

Table 29-3. WDT_SWCNR field descriptions

Field	Description
SWCN	Software watchdog count. The read-only WDT_SWCNR[SWCN] field reflects the current value in the watchdog counter. Writing to the WDT_SWCNR register has no effect, and write cycles are terminated normally. Reset initializes the WDT_SWCNR[SWCN] field to 0xFFFF. **Note:** Reading the 16 LS bits of 32-bit WDT_SWCNR register with two 8-bit reads is not guaranteed to return a coherent value. Always use a half-word or word access to read this register.

29.2.2.3 Software Watchdog Service Register (WDT_SWSRR)

The Software Watchdog Service Register (WDT_SWSRR) is shown in Figure 29-3. After the WDT is enabled, it must be periodically reset. This is accomplished in applications software by writing the special service sequence of 0x556C followed by 0xAA39 to the WDT_SWSRR register before the watchdog counter decrements to 0x0000. If the WDT_SWSRR register is not serviced before the timeout, the watchdog timer generates a system reset or MCP interrupt.

Both writes must occur before the timeout in the order listed; however, any number of instructions can be executed between the two writes. Writing any value other than 0x556C or 0xAA39 to the WDT_SWSRR register resets the servicing sequence. Both values must be written to keep the watchdog timer from decrementing to 0x0000. Reset initializes the WDT_SWSRR[WS] field to 0x0000. WDT_SWSRR can be written at any time, but returns all 0s when read.

Address: Base + 0x0E Access: User write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	WS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-3. Software Watchdog Service Register (WDT_SWSRR)

Table 29-4. WDT_SWSRR field descriptions

Field	Description
WS	Software watchdog service. The user must write 0x556C followed by 0xAA39 to this register to prevent WDT timeout. WDT_SWSRR[WS] can be written at any time, but returns all zeros when read.

29.3 Functional Description

29.3.1 Software Watchdog Timer Unit

To manage a case in which the software becomes trapped in a loop with no controlled exit, the MPC5125 platform provides the WDT option to prevent system lock. Watchdog timer operations are configured in the Software Watchdog Control Register (WDT_SWCRR).

By default, the WDT is enabled after reset to cause a hardware reset or non-maskable MCP interrupt if the WDT decrements to 0x0000. If the WDT is not needed, the user must clear the WDT_SWCRR[SWEN] bit to disable it. If used, the WDT requires a special service sequence to be executed periodically. Without this periodic servicing, it times out and issues a reset or a nonmaskable MCP interrupt, as programmed in WDT_SWCRR[SWRI]. After software writes to the SWRI bit, the state of SWEN cannot be changed.

The WDT service sequence consists of the following two steps:

1. Write 0x556C to WDT_SWSRR.
2. Write 0xAA39 to WDT_SWSRR.

This special service sequence reloads the WDT, and the timing process begins again. If a value other than 0x556C or 0xAA39 is written to the WDT_SWSRR, the entire sequence must start over. Although the writes must occur in the correct order before a timeout, any number of instructions can be executed between the writes. This allows interrupts and exceptions to occur between the two writes when necessary. Figure 29-4 shows a state diagram for the WDT.

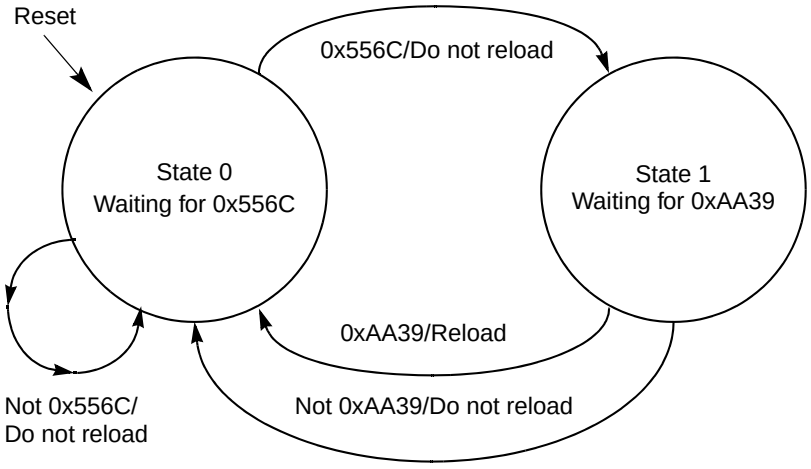


Figure 29-4. Software Watchdog Timer Service State Diagram

Although most software disciplines permit or even encourage the watchdog concept, some systems require a selection of timeout periods. For this reason, the WDT must provide a selectable range for the timeout period. Figure 29-5 shows how to manage this need.

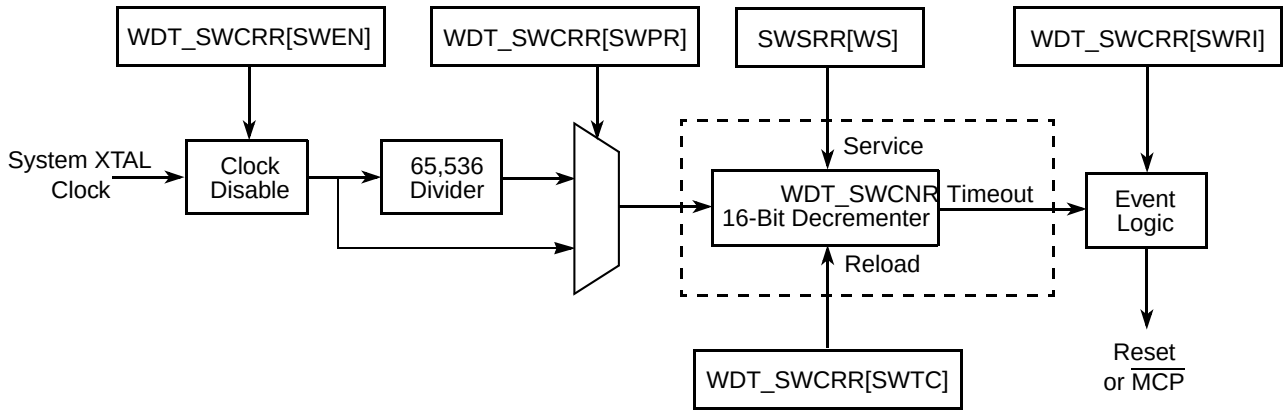


Figure 29-5. Software Watchdog Timer Functional Block Diagram

In Figure 29-5, the range is determined by the value of the WDT_SWCRR[SWTC] field. The value in SWTC is loaded into a 16-bit decrementer clocked by the system XTAL clock. An additional divide-by-65,536 prescaler value is used when needed.

The decrementer begins counting when loaded with a value from SWTC. After the timer reaches 0x000, a software watchdog expiration request is issued to the reset or MCP control logic. Upon reset, the SWTC field is set to the maximum value and is again loaded into the software watchdog count register (WDT_SWCRR), starting the process over. When a new value is loaded into SWTC, the WDT is not updated until the servicing sequence is written to the WDT_SWSRR. If WDT_SWCRR[SWEN] is loaded with 0, the modulus counter does not count.

29.3.2 Modes of Operation

The WDT unit can operate in the following modes:

- WDT enable/disable mode

  If the WDT is not needed, the user can disable it. The WDT_SWCRR[SWEN] bit enables the watchdog timer. It should be cleared by software after a system reset to disable the WDT. When it is disabled, the watchdog counter and prescaler counter are held in a stopped state.

  — WDT enable mode (WDT_SWCRR[SWEN] = 1)

  This is the default value after soft reset.

  — WDT disable mode (WDT_SWCRR[SWEN] = 0)

  If the WDT is not needed, the user must clear WDT_SWCRR[SWEN] to disable it.

- WDT reset/interrupt output mode

  Without periodic software servicing, the WDT times out and issues a reset or a nonmaskable MCP interrupt.

  According to WDT_SWCRR[SWRI] programming, WDT causes a hardware reset or MCP interrupt to the core.

— Reset mode (WDT_SWCRR[SWRI] = 1)

WDT causes a hardware reset.

— Interrupt mode (WDT_SWCRR[SWRI] = 0)

WDT causes an MCP interrupt to the core.

- WDT prescaled/non-prescaled clock mode

  The WDT counter clock can be prescaled by programming the WDT_SWCRR[SWPR] bit that controls the divide-by-65,536 of the WDT counter.

  — Prescale mode (CRR[SWPR] = 1)

  The WDT clock is prescaled.

  — Non-prescale mode (CRR[SWPR] = 0)

  The WDT clock is not prescaled.

29.3.2.1 WDT Enable/Disable Mode

By default, from the release of RESET, the WDT is enabled. If the WDT is not needed, the user can disable it by clearing the WDT_SWCRR[SWEN] bit. The WDT_SWCRR[SWEN] bit is cleared by the software after a system reset to disable the WDT. When it is disabled, the watchdog counter and prescaler counter are held in a stopped state. The WDT_SWCRR register is a write-once register; therefore, only software can disable the WDT. After it is disabled, it can only be re-enabled by resetting the system.

If the WDT is not needed, the user must clear the WDT_SWCRR[SWEN] bit to disable it.

29.3.2.2 WDT Reset/Interrupt Output Mode

Without periodic software servicing, the WDT times out and issues a reset or a nonmaskable MCP interrupt.

If the WDT_SWCRR[SWRI] bit is clear when the WDT times out, an MCP interrupt to the CPU core is created. If the WDT_SWCRR[SWRI] bit is set when the WDT times out, a hardware reset is created.

29.3.2.3 WDT Prescaled/Non-Prescaled Clock Mode

The WDT counter clock can be prescaled by 1 or 65,536, depending up the setting of the prescale bit of the WDT_SWCRR.

Prescale mode (WDT_SWCRR[SWPR] = 1 — The WDT counter clock uses a prescaler of 65,536.

Non-prescale mode (WDT_SWCRR[SWPR] = 0) — The WDT counter clock uses a prescaler of 1.

This page is intentionally left blank.

Chapter 30
SRAM Memory (MEM)

30.1 Introduction

This chapter describes the multi-port on-chip static RAM (SRAM). The MEM contains one bank of 32 KB SRAM. The MEM block interface contains three bus ports. The MEM block is implemented as a crossbar switch; Bus arbitration only occurs when two master bus ports try to access the same SRAM bank. The MEM block arbitrates between the bus ports with a fixed priority for each bus as follows: bus1 (highest priority), bus2 (2nd priority), and bus3 (lowest priority). The higher priority bus retains access to the SRAM as long as the bus maintains its request.

For the MPC5125, the MEM ports are configured as follows:

- bus1—connected to the CSB for PPC execution.
- bus2—connected to the NFC.
- bus3—connected to FEC1, FEC2, USB1, USB2, DMA2.

Figure 30-1 is a high-level block diagram of the memory block with its associated interfaces.

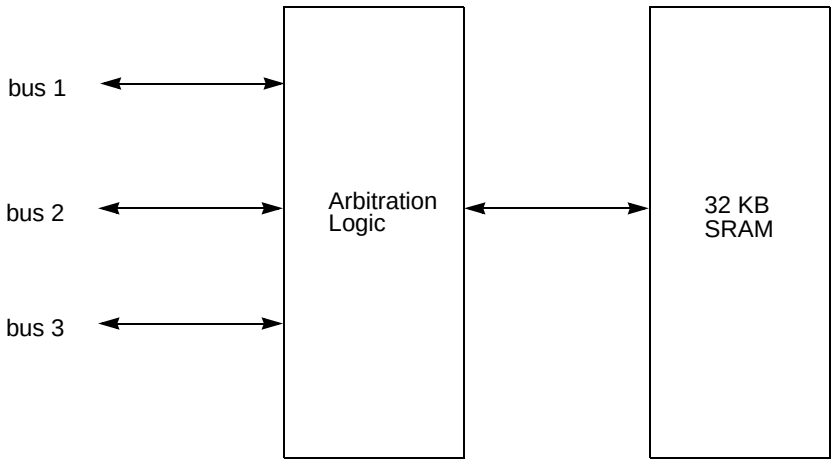


Figure 30-1. MEM Block Simplified Block Diagram

This page is intentionally left blank.

Chapter 31
Temperature Sensor

31.1 Introduction

The temperature sensor module monitors the internal temperature of the MPC5125. The temperature sensor module asserts two internal signals. The first signal is TEMP_105. The TEMP_105 signal asserts at one of eight programmable levels: 35 ºC, 45 ºC, 55 ºC, 65 ºC, 75 ºC, 85 ºC, 95 ºC, or 105 ºC. When the TEMP_105 signal asserts, an interrupt is signaled (if enabled) to the CPU core. The second signal, TEMP_125, is set to assert when the chip temperature reaches or exceeds 125 °C. When the TEMP_125 signal asserts, a machine check exception is signaled to the CPU core. In either case, some type of machine check exception handler or interrupt service routine must be called that starts reducing the power consumption of the device, possibly by shutting down various modules or setting clocks to lower frequencies, etc.

The temperature measurement accuracy is limited to ±5 °C.

31.1.1 Normal Operation Mode

From the release of HRESET, the temperature sensor module is enabled and running. The TEMP_105 and TEMP_125 signals are active. There are several registers that control the state and condition of the temperature sensor module. These registers are located in modules other than the temperature sensor module and are described in detail there.

The temperature sensor module can be enabled or powered down by use of the TEMPPD Bit in the Section 2.3.1.1.2, “System Priority Configuration Register (SPCR).” Setting the TEMPPD bit to a logic 0 enables the temperature sensor module. Setting TEMPPD bit to a logic 1 puts the temperature sensor module in a power down condition. Powering down the temperature sensor module prevents it from asserting either the TEMP_105 or TEMP_125 signals.

The TEMP_105 can be programmed to one of eight temperature trip levels. The trip temperature is programmed using the TEMPSEL bits in the c.

The TEMP_125 signal is programmed to assert when the chip temperature reaches or exceeds 125 °C. This temperature cannot be changed. When the TEMP_125 signal asserts, the TEMP125C bit sets in the Section 20.3.1.15, “System Error Status Register (SERSR).” If unmasked, the TEMP125C bit creates a machine check exception. The TEMP125C bit can be masked by the TEMP125C bit in the Section 20.3.1.16, “System Error Mask Register (SERMR).” The default state of the TEMP125C bit in the Section 20.3.1.16, “System Error Mask Register (SERMR),” is a logic 0 meaning that the TEMP_125C signal does not create a machine check exception. Setting the TEMP125C bit in the SERMR to a logic 1 allows the TEMP_125 signal to create a machine check exception.

The TEMP_105 signal is programmed to assert at one of eight programmable temperature levels. When the TEMP_105 signal asserts, the TEMP105C bit in the Section 20.3.1.3, “System Internal Interrupt Pending Registers (SIPNR_H and SIPNR_L),” asserts. If the TEMP105C bit in the SIPNR_L register asserts to a logic 1 and is not masked by the TEMP105C bit in the Section 20.3.1.8, “System Internal Interrupt Mask Register (SIMSR_H and SIMSR_L)” register, a normal interrupt is signaled to the CPU.

Chapter 32
Universal Serial Bus Interface with On-The-Go

32.1 Introduction

This chapter describes the universal serial bus (USB) interface of the MPC5125. The USB interface implements many industry standards. However, it is beyond the scope of this document to document the intricacies of these standards. Instead, it is left to the reader to refer to the governing specifications.

The following documents are available from the USB Implementers Forum web page at:
http://www.usb.org/developers/docs/

- *Universal Serial Bus Specification*, Rev. 2.0
- *On-The-Go Supplement to the USB 2.0 Specification*, Rev. 1.0a

The following documents are available from the Intel USB Specifications web page at:
http://www.usb.org/developers/docs/

- *Enhanced Host Controller Interface Specification for Universal Serial Bus*, Rev. 1.0
- *USB 2.0 Transceiver Macrocell Interface (UTMI) Specification*, Ver. 1.05

The following documents are available from the ULPI web page at:
http://www.usb.org/developers/docs/

- *UTMI+ Specification*, Rev. 1.0
- *UTMI Low Pin-Count Interface (ULPI) Specification*, Rev. 1.0

32.1.1 Overview

The MPC5125 implements two USB modules, having a dual-role (DR) or On-The-Go (OTG) capabilities. The USB1 and USB2 module can be connected to an external PHY using the ULPI protocol. Collectively, the modules and external ports are called the USB interface.

Throughout the document the term USB controller reflects both USB1/USB2 cores unless otherwise stated.

32.1.2 Features

The USB controller (USB1/USB2) supplies the following features:

- Supports USB device mode
- Supports USB On-The-Go mode including host capability
- Complies with USB specification Rev 2.0
- Supports high-speed (480 Mbit/s), full-speed (12 Mbit/s), and low-speed (1.5 Mbit/s) operations

- Supports external PHY with UTMI+ low pin count (ULPI) interface
- Supports operation as a standalone USB device
  - — Supports one upstream facing port
  - — Supports four programmable, bidirectional USB endpoints
- Host Mode supports direct connect of LS/FS devices

32.1.3 Modes of Operation

The USB1/USB2 has three basic operating modes: Host, Device, and On-the-Go (OTG).

Both USB modules require an external ULPI PHY.

32.2 Memory Map/Register Definitions

This section provides the memory map and detailed descriptions of all USB interface registers. Table 32-1 shows the memory map of the USB interface.

Table 32-1. USB memory map

Offset from USB_BASE[1] USB1: 0xFF40_3000 USB2: 0xFF40_4000	Register	Access[2]	Reset Value	Section/Page
0x000	USB_ID—Identification register	R	0xE241_FA05	32.2.1.1/32-817
0x004	USB_HWGENERAL—General hardware parameters register	R	0x0000_0085	32.2.1.2/32-818
0x008	USB_HWHOST—Host hardware parameters register	R	0x1002_0001	32.2.1.3/32-819
0x00C	USB_HWDEVICE—Device hardware parameters register	R	0x0000_0009	32.2.1.4/32-819
0x010	USB_HWTXBUF—TX buffer hardware parameters register	R	0x1007_0908	32.2.1.4/32-820
0x014	USB_HWRXBUF—RX buffer hardware parameters register	R	0x0000_0808	32.2.1.5/32-820
0x018–0x07F	Reserved			
0x080	USB_GPTIMER0LD—General Purpose Timer load value 0 register	R	0x0000_0000	32.2.2.5/32-826
0x084	USB_GPTIMER0CTRL—General Purpose Timer 0 Control register	R/W	0x0000_0000	32.2.2.5/32-826
0x088	USB_GPTIMER1LD—General Purpose Timer load value 1 register	R	0x0000_0000	32.2.2.5/32-826
0x08C	USB_GPTIMER1CTRL—General Purpose Timer 1 Control register	R/W	0x0000_0000	32.2.2.5/32-826
0x090	USB_SBUSCFG—System Bus Interface Control register	R/W	0x0000_0808	32.2.1.6/32-821
0x094–0x0FF	Reserved			
0x100	USB_CAPLENGTH—Capability register length / HCIVERSION—Host interface version number register	R	0x0100_0040	32.2.2.1/32-822

Table 32-1. USB memory map (continued)

Offset from USB_BASE[1] USB1: 0xFF40_3000 USB2: 0xFF40_4000	Register	Access[2]	Reset Value	Section/Page
0x104	USB_HCSPARAMS—Host control structural parameters register	R	0x0001_0011	32.2.2.2/32-823
0x108	USB_HCCPARAMS—Host Control Capability Parameters Register	R	0x0000_0006	32.2.2.3/32-824
0x10C–0x11F	Reserved			
0x120	USB_DCIVERSION—Device Interface Version Number Register	R	0x0000_0001	32.2.2.4/32-825
0x124	USB_DCCPARAMS—Device Controller Parameters Register	R	0x0000_0184	32.2.2.5/32-826
0x128–0x13F	Reserved			
0x140	USB_USBCMD—USB Command Register	R/W	0x0000_0000	32.2.4.1/32-828
0x144	USB_USBSTS—USB Status Register	R/W	0x0000_0000	32.2.4.2/32-831
0x148	USB_USBINTR—USB Interrupt Enable Register	R/W	0x0000_0000	32.2.4.3/32-832
0x14C	USB_FRINDEX—USB Frame Index Register	R/W	0x0000_0000	32.2.4.4/32-836
0x150	Reserved			
0x154[3]	USB_PERIODICLISTBASE—Periodic Frame List Base Address Register	R/W	0x0000_0000	32.2.4.6/32-837
0x154	USB_DEVICEADDR—Device Address Register	R/W	0x0000_0000	32.2.4.7/32-838
0x158	USB_ASYNCLISTADDR—Current Asynchronous List Address Register (Host Mode) (non-EHCI)	R/W	0x0000_0000	32.2.4.8/32-839
0x158	USB_ENDPOINTLISTADDR—Endpoint List Address Register (Device Mode) (non-EHCI)	R/W	0x0000_0000	32.2.4.9/32-840
0x15C	USB_TTCTRL—Host Controller Embedded TT Asynchronous Buffer Status Register	R/W	0x0000_0000	32.2.4.10/32-840
0x160	USB_BURSTSIZE—Master Interface Data Burst Size Register (non-EHCI)	R/W	0x0000_1010	32.2.4.11/32-841
0x164	USB_TXFILLTUNING—Transmit FIFO Tuning Controls Register (non-EHCI)	R/W	0x0000_0000	32.2.4.12/32-842
0x168–0x16F	Reserved			
0x170	USB_ULPIVIEWPORT—ULPI Viewport Register	R/W	0x0800_0000	32.2.4.13/32-843
0x174	Reserved			
0x178	USB_ENDPTNAK—Endpoint NAK Register	R/W	0x0000_0000	32.2.4.14/32-845
0x17C	USB_ENDPTNAKEN—Endpoint NAK Enable Register	R/W	0x0000_0000	32.2.4.15/32-846

Table 32-1. USB memory map (continued)

Offset from USB_BASE[1] USB1: 0xFF40_3000 USB2: 0xFF40_4000	Register	Access[2]	Reset Value	Section/Page
0x180	USB_CONFIGFLAG—Configured Flag Register	R	0x0000_0001	32.2.4.16/32-847
0x184	USB_PORTSC0—Port Status/Control Register 0	R/W	0x8C00_0000	32.2.4.17/32-847
0x188	USB_PORTSC1—Port Status/Control Register 1	R/W	0x8C00_0000	32.2.4.17/32-847
0x18C–0x1A0F	Reserved			
0x1A4	USB_OTGSC—On-the-Go Status and Control Register (non-EHCI)	R/W	0x0000_0000	32.2.4.18/32-852
0x1A8	USB_USBMODE—USB Device Mode Register (non-EHCI)	R/W	0x0000_0000	32.2.4.19/32-854
0x1AC	USB_ENDPTSETUPSTAT—Endpoint Setup Status Register (non-EHCI)	R/W	0x0000_0000	32.2.4.20/32-856
0x1B0	USB_ENDPTPRIME—Endpoint Initialization Register (non-EHCI)	R/W	0x0000_0000	32.2.4.21/32-856
0x1B4	USB_ENDPTFLUSH—Endpoint De-initialize Register (non-EHCI)	R/W	0x0000_0000	32.2.4.22/32-857
0x1B8	USB_ENDPTSTATUS—Endpoint Status Register (non-EHCI)	R	0x0000_0000	32.2.4.23/32-858
0x1BC	USB_ENDPTCOMPLETE—Endpoint Complete Register (non-EHCI)	R/W	0x0000_0000	32.2.4.24/32-859
0x01C0	USB_ENDPTCTRL0—Endpoint Control Register 0 (non-EHCI)	R/W	0x0080_0080	32.2.4.25/32-860
0x1C4	USB_ENDPTCTRL1—Endpoint Control Register 1 (non-EHCI)	R/W	0x0000_0000	32.2.4.26/32-861
0x1C8	USB_ENDPTCTRL2—Endpoint Control Register 2 (non-EHCI)	R/W	0x0000_0000	32.2.4.26/32-861
0x1CC	USB_ENDPTCTRL3—Endpoint Control Register 3 (non-EHCI)	R/W	0x0000_0000	32.2.4.26/32-861
0x1D0–0x1FF	Reserved			
0x200	USB_USBGENCTRL — USB General Control Register (non-EHCI)	R/W	0x0000_0000	32.2.4.27/32-863
0x204–0x3FF	Reserved			

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

2 In this column, R/W = Read/Write, R = Read-only, and W = Write-only.

3 In host mode, this address contains the USB_PERIODICLISTBASE register. in device mode, his address contains the USB_DEVICEADDR register.

The following sections provide details about the registers in the USB memory map.

Power Architecture		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Addr	Register																																
000	USB_ID	VersionID				Version				Revision				Tag				1	1	NID								0	0	ID			
004	USB_HW GENERAL	Reserved																					SM		PHYM			PHYW		BWT	CLKC		RT
008	USB_HW HOST	TTPER								TTASY								Reserved												NPORT			HC
00C	USB_HW DEVICE	Reserved																										DEVEP					DC
010	USB_HW TXBUF	TXCLR	Reserved							TXCHANADD								TXADD								TXBURST							
014	USB_HW RXBUF	Reserved																								RXADD				RXBURST			
080	USB_GP TIMER0LD									GPTLD																							
084	USB_GPT IMER0 CTRL	GPTRUN	GPTRST	Reserved					GPTMODE	GPTCNT																							
088	USB_GP TIMER1LD									GPTLD																							
08C	USB_GPT IMER1 CTRL	GPTRUN	GPTRST	Reserved					GPTMODE	GPTCNT																							
100	USB_CAP LENGTH/ HCIVERSION	HCIVERSION																Reserved								CAPLENGTH							
104	USB_HCS PARAMS	Reserved			N_TT				N_PPT				Reserved			PI		N_CC			N_PPC				Reserved		PPC	N_PORTS				
108	USB_HCC PARAMS	Reserved																EECP							IST				ASP	PFL	ADC	
120	USB_DCI VERSION	Reserved																DCIVERSION															
124	USB_DCC PARAMS	Reserved																							HC	DC	Reserved		DEN				
140	USB_USB CMD	Reserved								ITC								FS2	ATDTW	SUTW	R	ASPE		ASP			LR	IAA	ASE	PSE	FS		RST	RS
144	USB_USB STS	Reserved					TI1	TI0	Reserved			UPI	UAI			NAKI	AS	PS	RCL	HCH		ULPII		SLI	SRI	URI	AAI	SEI	FRI	PCI	UEI	UI	
148	USB_USB INTR	Reserved					TIE1	TIE0	Reserved			UPIA	UAIE			NAKE	Reserved					ULPIE		SLE	SRE	URE	AAE	SEE	FRE	PCE	UEE	UE	
14C	USB_FR INDEX	Reserved																					FRINDEX										
154[1]	USB_PERI ODICLIST BASE	PERBASE																					Reserved										

Figure 32-1. Summary of Register Layout

Power Architecture		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
154	DEVICEADDR	USBADR							USBADRA	Reserved																							
158	USB_ASYNCLISTADDR	ASYBASE																											Reserved				
158	USB_ENDPOINTLISTADDR	EPBASE																						Reserved									
15C	USB_TTCTRL	R	TTHA							Reserved																							
160	BURSTSIZE	Reserved																TXPBURST								RXPBURST							
164	USB_TXFILLTUNING	Reserved												TXFIFOTHRES				Reserved			TXSCHEALTH						TXSCHOH					
170	USB_ULPIVIEWPORT	ULPIWU	ULPIRUN	ULPIRW	R	ULPISS	ULPIPORT			ULPIADDR								ULPIDATARD								ULPIDATAWR							
178	USB_ENDPTNAK	Reserved												EPTN				Reserved											EPRN				
17C	USB_ENDPTNAKEN	Reserved												EPTNE				Reserved											EPRNE				
180	USB_CONFIGFLAG	Reserved																															1
184	USB_PORTSC1	PTS			PTW	PSPD			PFSC	PHCD	WKOC	WKDS	WKCN	PTC				PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
1A4	USB_OTGSC		DPIE		BSEIE	BSVIE	ASVIE	AVVIE	IDIE		DPIS	1mss	BSEIS	BSVIS	ASVIS	AVVIS	IDIS		DPS	1msT	BSE	BSV	ASV	AVV	ID	HABA	HADP	IDPU	DP	OT	HAAR	VC	VD
1A8	USB_USBMODE	Reserved																										VPBS	SDIS	SLOM	ES	CM	
1AC	USB_ENDPTSETUPSTAT	Reserved																												ENDPTSETUPSTAT			
1B0	USB_ENDPTPRIME	Reserved												PETB				Reserved											PERB				
1B4	USB_ENDPTFLUSH	Reserved												FETB				Reserved											FERB				
1B8	USB_ENDPTSTATUS	Reserved												ETBR				Reserved											ERBR				
1BC	USB_ENDPTCOMPLETE	Reserved												ETCE				Reserved											ERCE				
1C0	USB_ENDPTCTRL0	Reserved								TXE	Reserved			TXT			TXS	Reserved								RXE	Reserved			RXT			RXS
1C4	USB_ENDPTCTRL1	Reserved								TXE	TXR	TXI		TXT		TXD	TXS	Reserved								RXE	RXR	RXI		RXT		RXD	RXS
1C8	USB_ENDPTCTRL2	Reserved								TXE	TXR	TXI		TXT		TXD	TXS	Reserved								RXE	RXR	RXI		RXT		RXD	RXS
1CC	USB_ENDPTCTRL3	Reserved								TXE	TXR	TXI		TXT		TXD	TXS	Reserved								RXE	RXR	RXI		RXT		RXD	RXS

Figure 32-1. Summary of Register Layout (continued)

Power Architecture	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
200 USB_USBGENCTRL	Reserved																										WU_INT_CLR	ULPI_SEL	PPP	PFP	WU_ULPI_EN	WU_IE

[1] In host mode, this address contains the USB_PERIODICLISTBASE register. in device mode, his address contains the USB_DEVICEADDR register.

Figure 32-1. Summary of Register Layout (continued)

32.2.1 Module Identification Registers

The identification registers declare the slave interface presence and include hardware configuration parameters. These registers include the identification register, the general hardware parameters register, the host hardware parameters register, the device hardware parameters register, the Tx buffer hardware parameters register, and the Rx buffer hardware parameter register. These are not defined by the EHCI specification.

32.2.1.1 Identification (USB_ID) Register (Non-EHCI)

The Identification (USB_ID) register provides a simple way to determine if the module is present in the system. The USB_ID register identifies the module and its revision. Figure 32-2 shows the USB_ID register. Table 32-2 provides bit descriptions for the USB_ID register.

Address: Base + 0x000 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VERSIONID			VERSION				REVISION				TAG				
W																
Reset	1	1	1	0	0	0	1	0	0	1	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1	1	1	1	1	0	1	0	0	0	0	0	0	1	0	1

Figure 32-2. Identification (USB_ID) Register

Table 32-2. USB_ID field descriptions

Field	Description
VERSIONID	Internal version counter.
VERSION	Version of the USB core (<VERSION>,<REVISION><TAG>).

Table 32-2. USB_ID field descriptions (continued)

Field	Description
REVISION	Revision number of the USB core (<VERSION>,<REVISION><TAG>).
TAG	Tag of the USB core (<VERSION>,<REVISION><TAG>).
NID	Ones complement version of ID[5:0].
ID	Configuration number. This number is set to 0x05.

32.2.1.2 General Hardware Parameters (USB_HWGENERAL) Register (Non-EHCI)

The General Hardware Parameters (USB_HWGENERAL) register contains parameters that define the particular implementation of the module. Figure 32-3 shows the USB_HWGENERAL register. Table 32-3 provides bit descriptions for the USB_HWGENERAL register.

Address: Base + 0x004 (USB0, USB1) Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	SM		PHYM			PHYW		BWT	CLKC		RT
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1

Figure 32-3. General Hardware Parameters (USB_HWGENERAL) Register

Table 32-3. USB_HWGENERAL field descriptions

Field	Description
SM	SERIAL_MODE. Always 0 indicating no serial interface engine is present.
PHYM	PHY_MODE. The USB transceiver can only work with the ULPI interface, PHYM = 010.
PHYW	PHY Width. N/A. This field is only relevant for UTMI mode.
BWT	Reserved for manufacturing test. Reads as 0.
CLKC	Reserved for manufacturing test. Indicates the system and PHY clock partitioning of the design. Default 10.
RT	Reserved for manufacturing test. Always 1.

32.2.1.3 Host Hardware Parameters (USB_HWHOST) Register (Non-EHCI)

The Host Hardware Parameters (USB_HWHOST) register provides host hardware parameters for this implementation of the module. Figure 32-4 shows the USB_HWHOST register. Table 32-4 provides bit descriptions for the USB_HWHOST register.

Address: Base + 0x008 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TTPER								TTASY							
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	NPORT			HC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 32-4. Host Hardware Parameters (USB_HWHOST) Register

Table 32-4. USB_HWHOST field descriptions

Field	Description
TTPER	Transaction translator periodic contexts. The number of supported transaction translator periodic contexts. This always reads 0x10 (d'16).
TTASY	Transaction translator contexts. The number of transaction translator contexts.
NPORT	Always 0, indicating the number of ports available (NPORT + 1) for this host implementation.
HC	Always 1, indicating the module is host capable.

32.2.1.4 Device Hardware Parameters (USB_HWDEVICE) Register (Non-EHCI)

The Device Hardware Parameters (USB_HWDEVICE) register provides device hardware parameters for this implementation of the OTG module. Figure 32-5 shows the USB_HWDEVICE register. Table 32-5 provides bit descriptions for the USB_HWDEVICE register. This register is not defined in the EHCI specification.

Address: Base + 0x00C Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	DEVEP					DC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

Figure 32-5. Device Hardware Parameters (USB_HWDEVICE) Register

Table 32-5. USB_HWDEVICE field descriptions

Field	Description
DEVEP	Device endpoints. The number of supported endpoints; always 0x4.
DC	Always 1, indicating the OTG module is device capable.

32.2.1.4.1 Transmit Buffer Hardware Parameters (USB_HWTXBUF) Register (Non-EHCI)

The Transmit Buffer Hardware Parameters (USB_HWTXBUF) register provides the transmit buffer parameters for this implementation of the module. Figure 32-6 shows the USB_HWTXBUF register. Table 32-6 provides bit descriptions for the USB_HWTXBUF register.

Address: Base + 0x010 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXLCR	0	0	0	0	0	0	0	TXCHANADD							
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXADD								TXBURST							
W																
Reset	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0

Figure 32-6. Transmit Buffer Hardware Parameters (USB_HWTXBUF) Register

Table 32-6. USB_HWTXBUF field descriptions

Field	Description
TXLCR	Reserved. Always 1.
TXCHANADD	Transmit channel address. The number of address bits required to address one channel's worth of TX data. Always 0x7 (7).
TXADD	Transmit address. The number of address bits for the entire TX buffer. Always 0x9 (9).
TXBURST	Transmit burst. Indicates the number of data beats in a burst for transmit DMA data transfers. Always 0x8 (8).

32.2.1.5 Receive Buffer Hardware Parameters (USB_HWRXBUF) Register (Non-EHCI)

The Receive Buffer Hardware Parameters (USB_HWRXBUF) register provides the receive buffer parameters for this implementation of the module. Figure 32-7 shows the USB_HWRXBUF register. Table 32-7 provides bit descriptions for the USB_HWRXBUF register.

Address: Base + 0x014 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXADD								RXBURST							
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

Figure 32-7. Receive Buffer Hardware Parameters (USB_HWRXBUF) Register

Table 32-7. USB_HWRXBUF field descriptions

Field	Description
RXADD	Receive address. The number of address bits for the entire RX buffer. Always 0x8 (8).
RXBURST	Receive burst. Indicates the number of data beats in a burst for receive DMA data transfers. Always 0x8 (8).

32.2.1.6 System Bus Interface Configuration (USB_SBUSCFG) Register (Non-EHCI)

The System Bus Interface Configuration (USB_SBUSCFG) register contains the control for the system bus interface. Figure 32-8 shows the USB_SBUSCFG register. Table 32-8 provides bit descriptions for the USB_SBUSCFG register.

Address: Base + 0x090 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	BURSTMODE		
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

Figure 32-8. System Bus Interface Configuration (USB_SBUSCFG) Register

Table 32-8. USB_SBUSCFG field descriptions

Field	Description
BURSTMODE	Selects the following options for the burst signal of the Master Interface. 000 INCR burst of unspecified length. 001 INCR4, non-multiple transfers of INCR4 is decomposed into singles. 010 INCR8, non-multiple transfers of INCR8, is decomposed into INCR4 or singles. 011 INCR16, non-multiple transfers of INCR16, is decomposed into INCR8, INCR4 or singles. 100 Reserved, do not use. 101 INCR4, non-multiple transfers of INCR4 is decomposed into smaller unspecified length bursts. 110 INCR8, non-multiple transfers of INCR8 is decomposed into smaller unspecified length bursts. 111 INCR16, non-multiple transfers of INCR16 is decomposed into smaller unspecified length bursts. In all cases where the unspecified length burst is allowed, singles access may also occur, this is mostly true when the transaction is not 32-bit aligned. Two consecutive single accesses should not happen. When an INCR*x* burst size is selected and the transfer is not multiple of the INCR*x* burst, the burst is decomposed in the different ways. With BURSTMODE[2] = 1, the smaller bursts is unspecified length. with BURSTMODE[2] = 0, the smaller bursts are smaller INCR*x* or singles. For example, if it were required at a given time to transfer 22 words of information, for the following values of BURSTMODE the master sequences are: 101 INCR4 + INCR4 + INCR4 + INCR4 + INCR4 + INCR unspec. length. 110 INCR8 + INCR8 + INCR4 + INCR unspec. length. 111 INCR16 + INCR4 + INCR unspec. length. 001 INCR4 + INCR4 + INCR4 + INCR4 + INCR4 + SINGLE + SINGLE. 010 INCR8 + INCR8 + INCR4 + SINGLE + SINGLE. 011 INCR16 + INCR4 + SINGLE + SINGLE. When this field is different from zero, the values in the TXBURST/RXBURST bitfields in the USB_BURSTSIZE register are ignored by the controller. Internally the BURSTMODE is set to the value of the INCR*x* burst. Since this has a direct relation with the burst sizes you must be careful with AHB burst selected. Although the TXBURST/RXBURST are bypassed, this register can be written/read with no effect while the BURSTMODE field is non-zero. **Note:** Setting the BURSTMODE value to 000 might cause bus allocation during BULK or ISO transfers. **Note:** Changing this BURSTMODE field while a transaction is in progress yields undefined results. One possible way to prevent undefined results is to clear the Run/Stop (RS) bit in the USB_USBCMD register, after the HCHALTED is detected in USB_USBSTS.

32.2.2 Capability Registers

The capability registers specify the software limits, restrictions, and capabilities of the host/device controller implementation. The EHCI specification defines most of these registers. Registers not defined by the EHCI specification are noted in their descriptions.

32.2.2.1 Capability Registers Length (USB_CAPLENGTH) / Host Controller Interface Version (HCIVERSION) Register

This USB_CAPLENGTH register is an offset to add to the register base address to find the beginning of the operational register space, the location of the USB_USBCMD register. The HCIVERSION entry contains a BCD encoding of the EHCI revision number supported by this host controller. The

most-significant byte of the HCIVERSION represents a major revision and the least-significant byte is the minor revision.

Figure 32-9 shows the USB_CAPLENGTH / HCIVERSION register. Table 32-9 provides bit descriptions for the CAPLENGTH register.

Address: Base + 0x100 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HCIVERSION															
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CAPLENGTH							
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 32-9. Capability Registers Length (USB_CAPLENGTH)

Table 32-9. USB_CAPLENGTH field descriptions

Field	Description
HCIVERSION	EHCI revision number supported by this host.
CAPLENGTH	Capability registers length. Value is 0x40.

32.2.2.2 Host Controller Structural Parameters (USB_HCSPARAMS) Register (EHCI-Compliant)

The Host Controller Structural Parameters (USB_HCSPARAMS) register contains structural parameters such as the number of downstream ports. Figure 32-10 shows the USB_HCSPARAMS register. Table 32-10 provides bit descriptions for the USB_HCSPARAMS register.

Address: Base + 0x104 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	N_TT				N_PTT				0	0	0	PI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	N_CC				N_PCC				0	0	0	PPC	N_PORTS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Figure 32-10. Host Controller Structural Parameters (USB_HCSPARAMS) Register

Table 32-10. USB_HCSPARAMS field descriptions

Field	Description
N_TT	Number of transaction translators. This field always reads 0.
N_PTT	Ports per transaction translator. This field always reads 0.
PI	Port indicators. This bit indicates whether the ports support port indicator control. This field always reads 1. 0 1 The port status and control registers include a r/w field for controlling the state of the port indicator.
N_CC	Number of Companion Controllers. This field indicates the number of companion controllers associated with the OTG controller. This field always reads 0.
N_PCC	Number of Ports per Companion Controller. This field always reads 0.
PPC	Power Port Control. This bit indicates whether the host controller supports port power control.
N_PORTS	Number of Ports. This field indicates the number of physical downstream ports implemented for host applications. The value of this field determines how many port registers are addressable in the operational register.

32.2.2.3 Host Controller Capability Parameters (USB_HCCPARAMS) Register

The Host Controller Capability Parameters (USB_HCCPARAMS) register identifies multiple mode control (time-base bit functionality) addressing capability. Figure 32-11 shows the USB_HCCPARAMS register. Table 32-11 provides bit descriptions for the USB_HCCPARAMS register.

Address: Base + 0x108 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EECP								IST				0	ASP	PFL	ADC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 32-11. Host Controller Capability Parameters Register (USB_HCCPARAMS)

Table 32-11. USB_HCCPARAMS field descriptions

Field	Description
EECP	EHCI Extended Capabilities Pointer. This optional field indicates existence of a capabilities list. A value of 0x00 indicates no extended capabilities are implemented. A non-zero value in this register indicates the offset in PCI configuration space of the first EHCI extended capability. The pointer value must be 0x40 or greater if implemented to maintain the consistency of the PCI header defined for this class of device. This field always reads 0.
IST	Isochronous Scheduling Threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is 0, the value of the least significant three bits indicates the number of microframes a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a 1, host software assumes the host controller may cache an isochronous data structure for an entire frame. This field always reads 0.
ASP	Asynchronous Schedule Park Capability. This bit indicates if the host controller supports the park feature for high-speed queue heads in the asynchronous schedule. The feature can be disabled or enabled and set to a specific level by using the asynchronous schedule park mode enable and asynchronous schedule park mode count fields in the USB_USBCMD register. This field is always 1 (park feature supported).
PFL	Programmable Frame List Flag. This bit indicates system software can specify and use a frame list length less that 1024 elements. Frame list size is configured via the USB_USBCMD register frame list size field. The frame list must always be aligned on a 4K page boundary. This requirement ensures the frame list is always physically contiguous. This field always reads 1.
ADC	64-Bit Addressing Capability. This field always reads 0. Only 32-bit addressing is supported.

32.2.2.4 Device Controller Interface Version (USB_DCIVERSION) Register (Non-EHCI)

The Device Controller Interface Version (USB_DCIVERSION) register contains a BCD encoding of the device controller interface. The most-significant byte of the register represents a major revision and the least-significant byte is the minor revision. Figure 32-12 shows the USB_DCIVERSION register. Table 32-12 provides bit descriptions for the USB_DCIVERSION register. This register is not defined in the EHCI specification.

Address: Base + 0x120 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DCIVERSION															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 32-12. Device Controller Interface Version (USB_DCIVERSION) Register

Table 32-12. USB_DCIVERSION field descriptions

Field	Description
DCIVERSION	Device interface revision number is implementation-dependent. The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

32.2.2.5 Device Controller Capability Parameters (USB_DCCPARAMS) Register (Non-EHCI)

The Device Controller Capability Parameters (USB_DCCPARAMS) register describes the overall host/device capability of the OTG module. Figure 32-13 shows the USB_DCCPARAMS register. Table 32-13 provides bit descriptions for the USB_DCCPARAMS register. This register is not defined in the EHCI specification.

Address: Base + 0x124 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	HC	DC	0	0	DEN				
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0

Figure 32-13. Device Controller Capability Parameters (USB_DCCPARAMS)

Table 32-13. USB_DCCPARAMS field descriptions

Field	Description
HC	Host Capable. Always 1, indicating the OTG controller can operate as an EHCI compatible USB 2.0 host.
DC	Device Capable. Always 1, indicating the OTG controller can operate as an USB 2.0 device. 0 No device capability (host only). 1 Device capability.
DEN	Device Endpoint Number. This field indicates the number of endpoints built into the device controller. Always 0x4.

32.2.3 Device/Host Timer Registers (Non-EHCI)

The host/device controller drivers can measure time related activities using these timer registers. These registers are not part of the standard EHCI controller.

32.2.3.1 General Purpose Timer Load Register (USB_ GPTIMER*x*LD)

There are two timers available (USB_GPTIMER0LD and USB_GPTIMER1LD), each with its own control register (USB_GPTIMER0CTRL and USB_GPTIMER1CTRL).

Figure 32-14 shows the USB_GPTIMER*x*LD register. Table 32-14 provides bit descriptions for the USB_GPTIMER*x*LD register.

Address: Base + 0x080 (USB_GPTIMER0LD)
Base + 0x088 (USB_GPTIMER1LD)
Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-14. General Purpose Timer Load Register (USB_GPTIMER*x*LD)

Table 32-14. USB_GPTIMER*x*LD field descriptions

Field	Description
GPTLD	General Purpose Timer Load Value. This field is the value loaded into the GPTCNT countdown timer on a reset action. This value represents the time in microseconds minus 1 for the timer duration. **Note:** The maximum value is 0xFF_FFFF.

32.2.3.2 General Purpose Timer Control Register (GPTIMER*x*CTRL)

Figure 32-15 shows the USB_GPTIMER*x*CTRL register. Table 32-15 provides bit descriptions for the USB_GPTIMER*x*CTRL register.

This register contains the control for each timer and a data field can be queried to determine the running count value. The timer has a granularity of 1 us and can be programmed to a little over 16 seconds. There are two modes supported by the timer; the first is a one-shot, and the second is a looped count described in Table 32-16. When the timer counter value transitions to 0, an interrupt can be generated through the use of the timer interrupts in the USB_USBSTS and USB_USBINTR registers.

Address: Base + 0x84 (USB_GPTIMER0CTRL)
Base + 0x8C (USB_GPTIMER1CTRL)

Access: User read/write

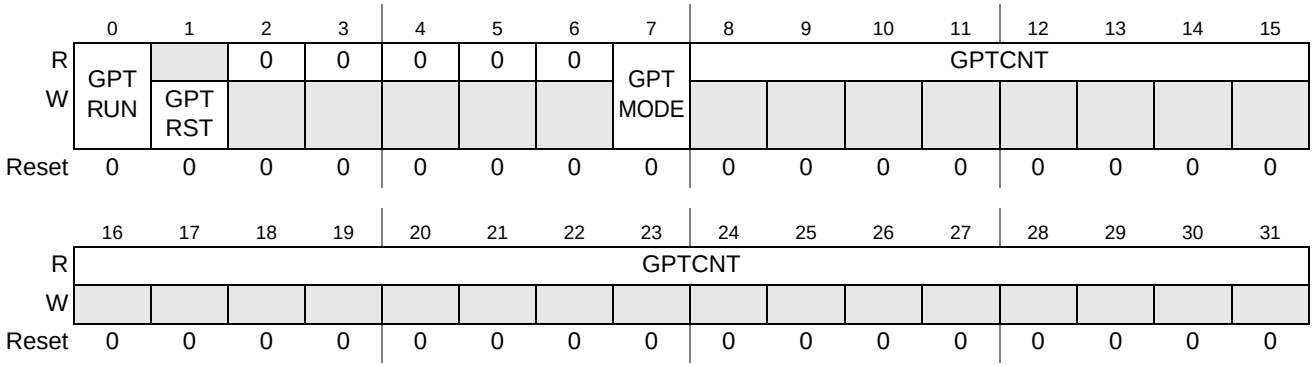


Figure 32-15. General Purpose Timer Control Register USB_GPTIMER*x*CTRL)

Table 32-15. USB_GPTIMER*x*CTRL field descriptions

Field	Description
GPTRUN	GPTRUN – General Purpose Timer Run. This bit enables the general purpose timer to run. Setting or clearing this bit does not have an effect on the GPTCNT value. 0 Timer Stop. 1 Timer Run.
GPTRST	GPTRST – General Purpose Timer Reset. Writing a 1 to this bit reloads the GPTCNT with the value in GPTLD. 0 No action. 1 Load counter.
GPTMODE	GPTMODE – General Purpose Timer Mode. This bit selects between a single timer countdown and a looped count down. In one-shot mode, the timer counts down to zero, generates an interrupt, and stops until the counter is reset by software. In repeat mode, the timer counts down to zero, generates an interrupt, and automatically reloads the counter to begin again.
GPTCNT	GPTCNT – General Purpose Timer Counter. Contains the value of the running timer.

32.2.4 Operational Registers

The operational registers are comprised of dynamic control or status registers that may be read-only, read/write, or read/write-1-to-clear. The following sections define the operational registers.

32.2.4.1 USB Command (USB_USBCMD) Register

The USB module executes the command indicated in this register.

Address: Base + 0x140 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	ITC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FS2	ATDTW	SUTW	0	ASPE	0	ASP		LR	IAA	ASE	PSE	FS1	FS0	RST	RS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-16. USB Command (USB_USBCMD) Register

Table 32-16. USB_USBCMD field descriptions

Field	Description
ITC	Interrupt Threshold Control. The system software uses this field to set the maximum rate at which the module issues interrupts. ITC contains the maximum interrupt interval measured in microframes. Valid values are shown below. 0x00 Immediate (no threshold) 0x01 1 microframe 0x02 2 microframes 0x04 4 microframes 0x08 8 microframes 0x10 16 microframes 0x20 32 microframes 0x40 40 microframes
FS2	See bit 3:2 below. This is a non-EHCI bit.
ATDTW	Add dTD TripWire. This is a non-EHCI bit is present on the OTG module only. This bit acts as a semaphore when a dTD is added to an active (primed) endpoint. This bit is set and cleared by software. This bit is also cleared by hardware when the state machine is in a hazard region where adding a dTD to a primed endpoint may go unrecognized. More information on the use of this bit is described in Section 32.8.10.1, “Device Operation,” of this manual.
SUTW	Setup TripWire. This is a non-EHCI bit present on the OTG module only. This bit acts as a semaphore when the 8 bytes of setup data read extracted from a QH by the DCD. If the setup lockout mode is off (See the USB_USBMODE register), a hazard exists when new setup data arrives and the DCD is copying setup from the QH for a previous setup packet. This bit is set and cleared by software or cleared by hardware. More information on the use of this bit is described in Section 32.8.10.1, “Device Operation,” of this manual.
ASPE	Asynchronous Schedule Park Mode Enable. Software uses this bit to enable or disable Park mode. 0 Disabled. 1 Enabled.
ASP	Asynchronous Schedule Park Mode Count. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the asynchronous schedule before continuing traversal of the asynchronous schedule. Valid values are 0x1 to 0x3. Software must not write a 0 to this field when park mode enable is a 1 as this results in undefined behavior.
LR	Light Host/Device Controller Reset (OPTIONAL). Not Implemented. Always 0.

Table 32-16. USB_USBCMD field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>IAA</td><td>Interrupt on Async Advance Doorbell. Software uses this bit as a doorbell to tell the controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell.
When the controller has evicted all appropriate cached schedule states, it sets the interrupt on async advance status bit in the USB_USBSTS register. If the interrupt on sync advance enable bit in the USB_USBINTR register is 1, the host controller asserts an interrupt at the next interrupt threshold.
The controller sets this bit to 0 after it has set the interrupt on sync advance status bit in the USB_USBSTS register to 1. Software should not write a 1 to this bit when the asynchronous schedule is inactive. Doing so yields undefined results.
This bit is only used in host mode. Writing a 1 to this bit when the OTG module is in device mode has undefined results.</td></tr>
<tr><td>ASE</td><td>Asynchronous Schedule Enable. This bit controls whether the controller skips processing the asynchronous schedule. Only used in host mode.
0 Do not process the asynchronous schedule.
1 Use the USB_ASYNCLISTADDR register to access the asynchronous schedule.</td></tr>
<tr><td>PSE</td><td>Periodic Schedule Enable. This bit controls whether the controller skips processing the periodic schedule. Only used in host mode.
0 Do not process the periodic schedule.
1 Use the USB_PERIODICLISTBASE register to access the periodic schedule.</td></tr>
<tr><td>FS[1:0]</td><td>Frame List Size. With bit 15, these bits make the FS[2:0] field. This field is read/write only if the programmable frame list flag (PFL bit) in the USB_HCCPARAMS register is set to 1. This field specifies the size of the frame list that controls which bits in the frame index register should be used for the frame list current index. Used only in host mode. Values below 256 elements are not defined in the EHCI specification.
<table>
<tr><th>FS2</th><th>FS1</th><th>FS0</th><th>Descriptions</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1024 elements (4096 bytes)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>512 elements (2048 bytes)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>256 elements (1024 bytes)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>128 elements (512 bytes)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>64 elements (256 bytes)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>32 elements (128 bytes)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>16 elements (64 bytes)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>8 elements (32 bytes)</td></tr>
</table></td></tr>
<tr><td>RST</td><td>Controller Reset. Software uses this bit to reset the controller. This bit is set to 0 by the controller when the reset process is complete. Software cannot terminate the reset process early by writing a 0 to this register.
Host Mode:
When software writes a 1 to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines, etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a 1 when the HCHalted bit in the USB_USBSTS register 0. Attempting to reset an actively running host controller results in undefined behavior.
Device Mode:
When software writes a 1 to this bit, the OTG controller resets its internal pipelines, timers, counters, state machines, etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. Writing a 1 to this bit in device mode is not recommended.</td></tr>
</table>

Table 32-16. USB_USBCMD field descriptions (continued)

Field	Description
RS	Run/Stop. **Host Mode:** When set to a 1, the controller proceeds with the execution of the schedule. The controller continues execution as long as this bit is set to 1. When this bit is set to 0, the host controller completes the current transaction on the USB and then halts. The HC halted bit in the status register indicates when the host controller has finished the transaction and has entered the stopped state. Software should not write a 1 to this field unless the controller is in the halted state (i.e., HCHalted in the USB_USBSTS register is a 1). **Device Mode:** Writing a 1 to this bit causes the OTG controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up becomes disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the OTG controller has been properly initialized. Writing a 0 to this causes a detach event. 0 Stop. 1 Run.

32.2.4.2 USB Status (USB_USBSTS) Register

The USB Status (USB_USBSTS) register indicates various states of each module and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus. Software clears certain bits in this register by writing a 1 to them.

Address: Base + 0x144 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	TI1	TI0	0	0	0	0	UPI	UAI	0	NAKI
W							w1c	w1c					w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AS	PS	RCL	HCH	0	ULPII	0	SLI	SRI	URI	AAI	SEI	FRI	PCI	UEI	UI
W						w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-17. USB Status (USB_USBSTS) Register

32.2.4.3 USB Interrupt Enable Register (USB_USBINTR)

Table 32-17. USB_USBSTS field descriptions

Field	Description
TI1	General Purpose Timer Interrupt 1 (GPTINT1). This bit is set when the counter in the USB_GPTIMER1CTRL register transitions to 0. Writing a 1 to this bit clears it.
TI0	General Purpose Timer Interrupt 0 (GPTINT0). This bit is set when the counter in the USB_GPTIMER0CTRL register transitions to 0. Writing a 1 to this bit clears it.
UPI	USB Host Periodic Interrupt (USBHSTPERINT). This bit is set by the host controller when the cause of an interrupt is a completion of a USB transaction where the transfer descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the host controller when a short packet is detected and the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and is always 0.
UAI	USB Host Asynchronous Interrupt (USBHSTASYNCINT). This bit is set by the host controller when the cause of an interrupt is a completion of a USB transaction where the transfer descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected and the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. The device controller does not use this bit and it is always 0.
NAKI	NAK Interrupt Bit. This bit is read-only. It is set by hardware when for a particular endpoint both the TX/RX endpoint NAK bit and the corresponding TX/RX endpoint NAK enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX endpoint NAK bits are cleared.
AS	Asynchronous Schedule Status. This bit reports the current real status of the asynchronous schedule. The controller is not required to immediately disable or enable the asynchronous schedule when software transitions the asynchronous schedule enable bit in the USB_USBCMD register. When this bit and the asynchronous schedule enable bit are the same value, the asynchronous schedule is enabled (1) or disabled (0). Used only in host mode. 0 Disabled. 1 Enabled.
PS	Periodic Schedule Status. This bit reports the current real status of the periodic schedule. The controller is not required to immediately disable or enable the periodic schedule when software transitions the periodic schedule enable bit in the USB_USBCMD register. When this bit and the periodic schedule enable bit are the same value, the periodic schedule is enabled (1) or disabled (0). Used only in host mode. 0 Disabled. 1 Enabled.
RCL	Reclamation. This is a status bit that detects an empty asynchronous schedule. Used only by the host mode. 0 Non-empty asynchronous schedule. 1 Empty asynchronous schedule.
HCH	Host Controller Halted. This bit is 0 when the run/stop bit is a 1. The controller sets this bit to 1 after it has stopped executing because of the run/stop bit being set to 0, by software or the host controller hardware (e.g., internal error). Used only in host mode. 0 Running. 1 Halted.
ULPII	ULPI Interrupt. When the ULPI Viewport is present in the design, an event completion sets this interrupt.

Table 32-17. USB_USBSTS field descriptions (continued)

Field	Description
SLI	DCSuspend. This is a non-EHCI bit present on the OTG module only. When a device controller enters a suspend state from an active state, this bit is set. This bit is only cleared by software writing a 1 to it. Only used by the device controller. 0 Active. 1 Suspended.
SRI	Host mode: This is a non-EHCI status bit. In host mode, this bit is set every 125 µs, provided the PHY clock is present and running (i.e., the port is NOT suspended). The host controller driver can use it as a time base. Device mode: SOF Received. When the OTG controller detects a Start Of (micro) Frame, this bit is be set to a 1. When a SOF is extremely late, the OTG controller automatically sets this bit to indicate an SOF was expected. Therefore, this bit is set roughly every 1 msec in device FS mode and every 125 msec in HS mode and is synchronized to the actual SOF received. Since the OTG controller is initialized to FS before connect, this bit is set at an interval of 1 msec during the prelude to the connect and chirp. Software writes a 1 to this bit to clear it.
URI	USB Reset Received. This is a non-EHCI bit present on the OTG module only. When the OTG controller detects a USB Reset and enters the default state, this bit is set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device mode. 0 No reset received. 1 Reset received.
AAI	Interrupt on Async Advance. System software can force the controller to issue an interrupt the next time the controller advances the asynchronous schedule by writing a 1 to the interrupt on async advance doorbell bit in the USB_USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host mode. 0 No async advance interrupt. 1 Async advance interrupt.
SEI	System Error. This bit is set when an error is detected on the system bus. If the system error enable (SEE) bit in the USB_USBINTR is set, an interrupt is generated. The interrupt and status bits remain asserted until cleared by writing a 1 to this bit. Additionally, when in host mode, the RUN/STOP (RS) bit of the USB_USBCMD register is cleared, effectively disabling the controller. For the OTG controller in device mode, an interrupt is generated, but no other action is taken. 0 Normal operation. 1 Error.
FRI	Frame List Rollover. The controller sets this bit to 1 when the frame list index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example, if the frame list size (as programmed in the frame list size field of the USB_USBCMD register) is 1024, the frame index register rolls over every time USB_FRINDEX [13] toggles. Similarly, if the size is 512, the controller sets this bit to a 1 every time FHINDEX [12] toggles. Used only by the host mode.
PCI	Host mode: Port Change Detect. The controller sets this bit to 1 when on any port a connect status occurs, a port enable/disable change occurs, or the force port resume bit is set as the result of a J-K transition on the suspended port. Device mode: The OTG controller sets this bit to 1 when it enters the full or high-speed operational state. When the USB exits full or high-speed operation states due to reset or suspend events, the notification mechanisms are the USB reset received bit and the DCSuspend bits respectively. The device controller detects resume signalling only. This bit is not EHCI compatible.

Table 32-17. USB_USBSTS field descriptions (continued)

Field	Description
UEI (USBERRINT)	USB Error Interrupt (USBERRINT). When completion of a USB transaction results in an error condition, this bit is set by the controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. See Table 32-97 in this chapter for more information on device error matrix. All others are ignored. 0 No error. 1 Error detected.
UI (USBINT)	USB Interrupt (USBINT). This bit is set by the controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt is active. The USB_USBSTS register continues to show interrupt sources even if the USB_USBINTR register disables them, allowing polling of interrupt events by the software.

Address: Base + 0x148 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	TIE1	TIE0	0	0	0	0	UPIA	UAIE	0	NAKE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	ULPIE	0	SLE	SRE	URE	AAE	SEE	FRE	PCE	UEE	UE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-18. USB Interrupt Enable Register (USB_USBINTR)

Table 32-18. USB_USBINTR field descriptions

Field	Description
TIE1	General Purpose Timer Interrupt Enable 1 When this bit is a 1 and the GPTINT1 bit in the USB_USBSTS register is also a 1, the controller issues an interrupt. The interrupt is acknowledged by software clearing the GPTINT1 bit.
TIE0	General Purpose Timer Interrupt Enable 0 When this bit is a 1 and the GPTINT0 bit in the USB_USBSTS register is also a 1, the controller issues an interrupt. The interrupt is acknowledged by software clearing the GPTINT0 bit.
UPIA	USB Host Periodic Interrupt Enable When this bit is a 1 and the USBHSTPERINT bit in the USB_USBSTS register is also a 1, the host controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHSTPERINT bit.
UAIE	USB Host Async. Interrupt Enable When this bit is a 1 and the USBHSTASYNCINT bit in the USB_USBSTS register is also a 1, the host controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHSTASYNCINT bit.

Table 32-18. USB_USBINTR field descriptions (continued)

Field	Description
NAKE	NAK Interrupt Enable Software sets this bit if it wants to enable the hardware interrupt for the NAK Interrupt bit. If this bit and the corresponding NAK Interrupt bit are set, a hardware interrupt is generated.
ULPIE	ULPI Enable When this bit is a 1 and the ULPI interrupt bit in the USB_USBSTS register transitions, the controller issues an interrupt. The interrupt is acknowledged by software writing a 1 to the ULPI Interrupt bit. Used by both host and device controller.
SLE	Sleep Enable. This is a non-EHCI bit. When this bit is a 1 and the DCSuspend bit in the USB_USBSTS register transitions, the OTG controller issues an interrupt. The interrupt is acknowledged by software writing a 1 to the DCSuspend bit. Used only in device mode. 0 Disabled. 1 Enabled.
SRE	SOF Received Enable. This is a non-EHCI bit. When this bit is a 1 and the SOF Received bit in the USB_USBSTS register is a 1, the controller issues an interrupt. The interrupt is acknowledged by software clearing the SOF received bit. 0 Disabled. 1 Enabled.
URE	USB Reset Enable. This is a non-EHCI bit present on the OTG module only. When this bit is a 1 and the USB reset received bit in the USB_USBSTS register is a 1, the device controller issues an interrupt. The interrupt is acknowledged by software clearing the USB reset received bit. Used only in device mode. 0 Disabled. 1 Enabled.
AAE	Interrupt on Async Advance Enable. When this bit is a 1 and the interrupt on async advance bit in the USB_USBSTS register is a 1, the controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the interrupt on async advance bit. Used only in host mode. 0 Disabled. 1 Enabled.
SEE	System Error Enable. When this bit is a 1 and the system error bit in the USB_USBSTS register is a 1, the controller issues an interrupt. The interrupt is acknowledged by software clearing the system error bit. 0 Disabled. 1 Enabled.
FRE	Frame List Rollover Enable. When this bit is a 1 and the frame list rollover bit in the USB_USBSTS register is a 1, the controller issues an interrupt. The interrupt is acknowledged by software clearing the frame list rollover bit. Used only by the host mode. 0 Disabled. 1 Enabled.
PCE	Port Change Detect Enable. When this bit is a 1 and the port change detect bit in the USB_USBSTS register is a 1, the controller issues an interrupt. The interrupt is acknowledged by software clearing the port change detect bit. 0 Disabled. 1 Enabled.
UEE	USB Error Interrupt Enable. When this bit is a 1 and the USBERRINT bit in the USB_USBSTS register is a 1, the controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBERRINT bit in the USB_USBSTS register. 0 Disabled. 1 Enabled.

Table 32-18. USB_USBINTR field descriptions (continued)

Field	Description
UE	USB Interrupt Enable. When this bit is a 1 and the USBINT bit in the USB_USBSTS register is a 1, the OTG controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBINT bit. 0 Disabled. 1 Enabled.

32.2.4.4 USB Frame Index (USB_FRINDEX) Register

In host mode, the controller uses the USB Frame Index (USB_FRINDEX) register to index the periodic frame list. The register updates every 125 microseconds (once each microframe). Bits [N–3] select a particular entry in the periodic frame list during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the frame list size field in the USB_USBCMD register.

This register must be written as a 32-bit word. Byte writes produce-undefined results. This register cannot be written unless the OTG controller is in the halted state as indicated by the HCHalted bit. A write to this register while the run/stop hit is set to a 1 produces undefined results. Writes to this register also affect the SOF value.

In device mode, this register is read-only, and the OTG controller updates the USB_FRINDEX[13–3] register from the frame number indicated by the SOF marker. When a SOF is received by the USB bus, USB_FRINDEX[13–3] is checked against the SOF marker. If USB_FRINDEX[13–3] is different from the SOF marker, USB_FRINDEX[13–3] is set to the SOF value and USB_FRINDEX[2–0] is set to 0 (that is, SOF for 1 msec frame). If USB_FRINDEX[13–3] is equal to the SOF value, USB_FRINDEX[2–0] is incremented (that is, SOF for 125 µsec microframe.)

Table 32-19 illustrates values of N based on the value of the frame list size in the USB_USBCMD register when used in host mode.

Table 32-19. USB_FRINDEX N Values

USB_USBCMD[FS]	Frame List Size	USB_FRINDEX N Value
000	1024 elements (4096 bytes)	12
001	512 elements (2048 bytes)	11
010	256 elements (1024 bytes)	10
011	128 elements (512 bytes)	9
100	64 elements (256 bytes)	8
101	32 elements (128 bytes)	7
110	16 elements (64 bytes)	6
111	8 elements (32 bytes)	5

Address: Base + 0x14C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	FRINDEX													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-19. USB Frame Index (USB_FRINDEX) Register

Table 32-20. USB_FRINDEX field descriptions

Field	Description
FRINDEX	Frame index. The value in this register increments at the end of each time frame (for example, microframe). Bits [N– 3] are used for the frame list current index. This means that each location of the frame list is accessed 8 times (frames or microframes) before moving to the next index. The value is the current frame number of the last frame transmitted. It is not used as an index. Bits 2–0 indicate the current microframe.

32.2.4.5 Control Data Structure Segment Register (CTRLDSSEGMENT)

The Control Data Structure Segment (CTRLDSSEGMENT) register is not implemented on the MPC5125.

32.2.4.6 Periodic Frame List Base Address (USB_PERIODICLISTBASE) Register

The Periodic Frame List Base Address (USB_PERIODICLISTBASE) register contains the beginning address of the Periodic Frame List in the system memory. The host controller driver loads this register prior to starting the schedule execution by the controller. The memory structure referenced by this physical memory pointer is assumed to be 4 KB aligned. The contents of this register are combined with the USB_FRINDEX register to enable the controller to step through the periodic frame list in sequence.

On the OTG module, this register is shared between the host and device mode functions. In host mode, it is the USB_PERIODICLISTBASE register; in device mode, it is the USB_DEVICEADDR register. See Section 32.2.4.7, “Device Address (USB_DEVICEADDR) Register (Non-EHCI),” for more information.

Address: Base + 0x154 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PERBASE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PERBASE (cont.)				0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-20. Periodic Frame List Base Address (USB_PERIODICLISTBASE) Register

Table 32-21. USB_PERIODICLISTBASE field descriptions

Field	Description
PERBASE	Base Address. These bits correspond to memory address signal [31:12]. Only used in the host mode.

32.2.4.7 Device Address (USB_DEVICEADDR) Register (Non-EHCI)

For the OTG module in device mode, the upper 7 bits of the Device Address (USB_DEVICEADDR) register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address matches all incoming addresses. Software reprograms the address after receiving a SET_ADDRESS descriptor.

The USBADRA can accelerate the SET_ADDRESS sequence by allowing the DCD to preset the USBADR register before the status phase of the SET_ADDRESS descriptor.

On the OTG module, this register is shared between the host and device mode functions. In device mode, it is the USB_DEVICEADDR register; in host mode, it is the USB_PERIODICLISTBASE register. See Section 32.2.4.6, “Periodic Frame List Base Address (USB_PERIODICLISTBASE) Register,” for more information. This register is not defined in the EHCI specification.

Address: Base + 0x154 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	USBADR							USBADRA	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-21. Device Address (USB_DEVICEADDR) Register

Table 32-22. USB_DEVICEADDR field descriptions

Field	Description
USBADR	Device Address. This field corresponds to the USB device address.
USBADRA	Device Address Advance. Default = 00. When this bit is 0, any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR is loaded from the holding register. Hardware automatically clears this bit on the following conditions: 01)IN is ACKed to endpoint 0. (USBADR is updated from staging register). 10)OUT/SETUP occur to endpoint 0. (USBADR is not updated). 11)Device Reset occurs (USBADR is reset to 0). **Note:** After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism ensures this specification is met when the DCD cannot write to the device address within 2 ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA equaling 1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR is programmed instantly at the correct time and meets the 2 ms USB requirement.

32.2.4.8 Current Asynchronous List Address Register (USB_ASYNCLISTADDR)

The Current Asynchronous List Address Register (USB_ASYNCLISTADDR) contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and always return 0s when read.

On the OTG module, this register is shared between the host and device mode functions. In host mode, it is the USB_ASYNCLISTADDR register; in device mode, it is the USB_ENDPOINTLISTADDR register. See Section 32.2.4.9, “Endpoint List Address (USB_ENDPOINTLISTADDR) Register (Non-EHCI),” for more information.

Address: Base + 0x158 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ASYBASE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ASYBASE (cont.)											0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-22. Current Asynchronous List Address Register (USB_ASYNCLISTADDR)

Table 32-23. USB_ASYNCLISTADDR field descriptions

Field	Description
ASYBASE	Link Pointer Low (LPL). These bits correspond to memory address signal [31:5]. This field may only reference a queue head (QH). Only used by the host controller.

32.2.4.9 Endpoint List Address (USB_ENDPOINTLISTADDR) Register (Non-EHCI)

For the OTG module in device mode, the Endpoint List Address (USB_ENDPOINTLISTADDR) register contains the address of the top of the endpoint list in system memory. Bits [10–0] of this register cannot be modified by the system software and always return 0s when read. The memory structure referenced by this physical memory pointer is assumed to be 64 bytes. The queue head is actually a 48-byte structure, but must be aligned on a 64-byte boundary. However, the USB_ENDPOINTLISTADDR[EPBASE] has a granularity of 2 KB, so in practice the queue head should be 2-KB aligned.

On the OTG module, this register is shared between the host and device mode functions. In device mode, it is the USB_ENDPOINTLISTADDR register; in host mode, it is the USB_ASYNCLISTADDR register. See Section 32.2.4.8, “Current Asynchronous List Address Register (USB_ASYNCLISTADDR),” for more information. This register is not defined in the EHCI specification.

Address: Base + 0x158 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EPBASE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EPBASE(cont.)					0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-23. Endpoint List Address (USB_ENDPOINTLISTADDR) Register

Table 32-24. USB_ENDPOINTLISTADDR field descriptions

Field	Description
EPBASE	Endpoint List Address. Address of the top of the endpoint list.

32.2.4.10 Host Controller Embedded TT Asynchronous Buffer Status (USB_TTCTRL) Register

The Host Controller Embedded TT Asynchronous Buffer Status (USB_TTCTRL) register contains parameters for internal TT operations. This register is not used in device controller operation.

Address: Base + 0x15C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TTHA							0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-24. Host Controller Embedded TT Asynchronous Buffer Status (USB_TTCTRL) Register

Table 32-25. USB_TTCTRL field descriptions

Field	Description
TTHA	Internal TT Hub Address Representation. Default is 0 (read/write). This field matches against the hub address field in QH and siTD to determine if the packet is routed to the internal TT for directly attached FS/LS devices. If the hub address in the QH or siTD does not match this address, the packet is broadcast on the high speed ports destined for a downstream high speed hub with the address in the QH/siTD.

32.2.4.11 Master Interface Data Burst Size (USB_BURSTSIZE) Register (Non-EHCI)

The Master Interface Data Burst Size (USB_BURSTSIZE) register controls and dynamically changes the burst size used during data movement on the initiator (master) interface. This register is not defined in the EHCI specification.

Address: Base + 0x160 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXPBURST								RXPBURST							
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0

Figure 32-25. Master Interface Data Burst Size (USB_BURSTSIZE) Register

Table 32-26. USB_BURSTSIZE field descriptions

Field	Description
TXPBURST	Programmable TX Burst Length. This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus. If the BURSTSIZE bitfield of the USB_SBUSCFG register is non-zero, the TXPBURST bitfield returns the value of the INCR*x* length.
RXPBURST	Programmable RX Burst Length. This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory. If the BURSTSIZE bitfield of the USB_SBUSCFG register is non-zero, the RXPBURST bitfield returns the value of the INCR*x* length.

32.2.4.12 Transmit FIFO Tuning Controls (USB_TXFILLTUNING) Register (Non-EHCI)

The Transmit FIFO Tuning Controls (USB_TXFILLTUNING) register controls and dynamically changes the burst size used during data movement on device DMA transfers. It is only used in host mode.

The fields in this register control performance tuning associated with how the module posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include how much data to post into the FIFO and an estimate for how long that operation should take in the target system.

Definitions:

T_0 = Standard packet overhead

T_1 = Time to send data payload

T_s = Total Packet Flight Time (send-only) packet ($T_s = T_0 + T_1$)

T_{ff} = Time to fetch packet into TX FIFO up to specified level.

T_p = Total Packet Time (fetch and send) packet ($T_p = T_{ff} + T_s$)

Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before end of the [micro]frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining in the [micro]frame is < T_s, packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the module eventually recovers, a mark is made in the scheduler health counter to show the occurrence of a back-off event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that begins after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and should be minimized (not necessarily eliminated). Use of the TSCHHEALTH (T_{ff}) parameter described below minimizes back-offs. This register is not defined in the EHCI specification.

Address: Base + 0x164 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	TXFIFOTHRES					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	TXSCHHEALTH					0	TXSCHOH						
W				w1c												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-26. Transmit FIFO Tuning Controls (USB_TXFILLTUNING) Register

Table 32-27. USB_TXFILLTUNING field descriptions

Field	Description
TXFIFOTHRES	FIFO Burst Threshold. This register controls the number of data bursts posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is two and this value should be as low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the stream disable bit in USB_USBMODE register is set.
TXSCHHEALTH	Scheduler Health Counter. These bits increment when the OTG controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register clears the counter.
TXSCHOH	Scheduler Overhead. These bits add an additional fixed offset to the schedule time estimator described above as T_{ff}. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267 μs when a device is connected in high-speed mode. The time unit represented in this register is 6.333 μs when a device is connected in low/full-speed mode. For most applications, TXSCHOH can be set to 4 or less. A good value to begin with is: TXFIFOTHRES × (BURSTSIZE × 4 bytes-per-word)/(40 × *TimeUnit*), always rounded to the next higher integer. *TimeUnit* is either 1.267 or 6.333 as noted earlier in this description. For example, if TXFIFOTHRES is 5 and BURSTSIZE is 8, set TXSCHOH to 5 × (8 × 4)/(40 × 1.267) = 4 for a high-speed link. If this value of TXSCHOH results in a TXSCHHEALTH count of 0 per second, try lowering the value by 1 if optimizing performance is desired. If TXSCHHEALTH exceeds 10 per second, try raising the value by 1. If streaming mode is disabled via the USB_USBMODE register, treat TXFIFOTHRES as the maximum value for purposes of the TXSCHOH calculation.

32.2.4.13 ULPI Viewport (USB_ULPIVIEWPORT) Register

The ULPI Viewport (USB_ULPIVIEWPORT) register provides indirect access to the ULPI PHY register set. Although the core performs access to the ULPI PHY register set, extraordinary circumstances may exist where software may need direct access.

CAUTION

Writes to the ULPI through the viewport can substantially harm standard USB operations. Read operations should have no harmful side effects to standard USB operations.

Two operations, wake-up and read/write, can be performed with the ULPI Viewport. Wake-up operation puts the ULPI interface into normal operation mode and re-enables the clock if necessary. A wakeup operation is required before accessing the registers when ULPI interface is operating in low-power mode, serial mode, or carkit mode. The ULPI state can be determined by reading the sync. state bit (ULPISS). If this bit is a 1, ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPISS indicates a 0, read/write operations are not able to execute. Undefined behavior results if ULPISS equals 0 and a read or write operation is performed. To execute a wakeup operation, write all 32 bits of the ULPI Viewport where ULPIPORT is constructed appropriately and the ULPIWU bit is 1 and ULPIRUN bit is 0. Poll the ULPI viewport until ULPIWU is 0 for the operation to complete.

To execute a read or write operation, write all 32 bits of the ULPI Viewport where ULPIDATWR, ULPIADDR, ULPIPORT, and ULPIRW are constructed appropriately and the ULPIRUN bit is 1. Poll the ULPI Viewport until ULPIRUN is 0 for the operation to complete. After ULPIRUN is 0, ULPIDATRD is valid if the operation was a read.

The polling method above could also be replaced by an interrupt driven routine using the ULPI interrupt defined in the USB_USBSTS and USB_USBINTR registers. When a wakeup or read/write operation complete, the ULPI interrupt is set.

There are several optional features that system software may need to enable or disable as part of system configuration. These bits are contained in the interface and OTG control registers of the ULPI PHY register set. These registers also contain bits controlled by the link dynamically and therefore should only be modified by system software using the set/clear access method. Direct writes to these registers could have harmful side effects to the standard USB operations. The optional bits are as follows: bits 3 through 7 in the interface control register and bits 6 and 7 in the OTG control register. Refer to the ULPI Specification Revision 1.1 for further information on the use of the optional features.

Address: Base + 0x170 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ULPI WU	ULPI RUN	ULPI RW	R	ULPI SS	ULPIPORT			ULPIADDR							
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ULPIDARD								ULPIDATWR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-27. ULPI Viewport (USB_ULPIVIEWPORT) Register

Table 32-28. USB_ULPIVIEWPORT field descriptions

Field	Description
ULPIWU	ULPI Wakeup. Writing the 1 to this bit begins the wakeup operation. The bit automatically transitions to 0 after the wakeup is complete. After this bit is set, the driver cannot set it back to 0. **Note:** The driver must never execute a wake-up and a read/write operation at the same time.
ULPIRUN	ULPI Read/Write Run. Writing the 1 to this bit begins the read/write operation. The bit automatically transitions to 0 after the read/write is complete. After this bit is set, the driver cannot set it back to 0. **Note:** The driver must never execute a wake-up and a read/write operation at the same time.
ULPIRW	ULPI Read/Write Control. This bit selects between running a read or a write operation 0 Read. 1 Write.
ULPISS	ULPI Sync State. This bit represents the state of the ULPI interface. Before reading this bit, the ULPIPORT field should be set accordingly if used with the multi-port host. Otherwise, this field should always remain 0. 0 In another state (e.g., carkit, low power). 1 Normal sync state.
ULPIPORT	ULPI Port Number. For the wakeup or read/write operation to be executed, this value selects the port number the ULPI PHY is attached to in the multi-port host. The range is 0 to 7. This field should always be written as a 0 for the non-multi port products.
ULPIADDR	ULPI Data Address. When a read or write operation is commanded, the address of the operation is written to this field.
ULPIDATRD	ULPI Data Read Value. After a read operation completes, the result is placed in this field.
ULPIDATWR	ULPI Data Write Value. When a write operation is commanded, the data to be sent is written to this field.

32.2.4.14 Endpoint NAK (USB_ENDPTNAK) Register

Address: Base + 0x178 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	EPTN			
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	EPRN			
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-28. Endpoint NAK (USB_ENDPTNAK) Register

Table 32-29. USB_ENDPTNAK field descriptions

Field	Description
EPTN	TX Endpoint NAK—Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. EPTN[3]—Endpoint #3 EPTN[2]—Endpoint #2 EPTN[1]—Endpoint #1 EPTN[0]—Endpoint #0
EPRN	RX Endpoint NAK—Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. EPRN[3]—Endpoint #3 EPRN[2]—Endpoint #2 EPRN[1]—Endpoint #1 EPRN[0]—Endpoint #0

32.2.4.15 Endpoint NAK Enable (USB_ENDPTNAKEN) Register

Address: Base + 0x17C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	EPTNE			
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	EPRNE			
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-29. Endpoint NAK Enable (USB_ENDPTNAKEN) Register

Table 32-30. USB_ENDPTNAKEN field descriptions

Field	Description
EPTNE	TX Endpoint NAK—Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. EPTNE[3]—Endpoint #3 EPTNE[2]—Endpoint #2 EPTNE[1]—Endpoint #1 EPTNE[0]—Endpoint #0
EPRNE	RX Endpoint NAK—Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. EPRNE[3]—Endpoint #3 EPRNE[2]—Endpoint #2 EPRNE[1]—Endpoint #1 EPRNE[0]—Endpoint #0

32.2.4.16 Configure Flag (USB_CONFIGFLAG) Register

A read from the Configure Flag (USB_CONFIGFLAG) register returns a constant of 0x0000_0001 to indicate all port routings default to this host controller. This EHCI register is not used in this implementation.

Address: Base + 0x180 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 32-30. Configure Flag (USB_CONFIGFLAG) Register

Table 32-31. USB_CONFIGFLAG field descriptions

Field	Description
—	Reserved. (0x0000_0001, all port routings default to this host)

32.2.4.17 Port Status and Control Registers (PORTSC*n*)

The OTG module has one port status and control register. The number of port registers implemented by a particular instantiation of a host controller is documented in the USB_HCSPARAMS register. Software uses this information as an input parameter to determine how many ports need service. This register is only reset when power is initially applied or in response to a controller reset. The initial conditions of a port are:

- No device connected
- Port disabled

If the port has port power control, this state remains until software applies power to the port by setting port power bit to one.

For the OTG module in device mode, the OTG controller does not support power control. Port control in device mode is only for status port reset, suspend, and current connect status. It also initiates test mode or forces signaling and allows software to put the PHY into low-power suspend mode and disable the PHY clock.

Address: Base + 0x184 (USB_PORTSC0, USB0)
Base + 0x188 (USB_PORTSC1, USB1)

Access: User read/write

Base + 184h + (4 × (Port Number –1))
USB1/USB2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PTS		0	0	PSPD		0	PFSC	PHCD	WKOC	WKDS	WKCN	PTC			
W																
Reset	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
W											w1c		w1c		w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-31. Port Status and Control *n* (USB_PORTSC*n*) Register

Table 32-32. USB_PORTSC*n* field descriptions

Field	Description
PTS	Port Transceiver Select. This register bit controls which parallel transceiver interface is selected. 10 ULPI parallel interface All other values are reserved.
PSPD	Port Speed. This read-only register field indicates the speed at which the port is operating. This bit is not defined in the EHCI specification. 00 Full-speed 01 Low-speed 10 High-speed 11 Undefined
PFSC	Port Force Full-speed Connect. This bit disables the chirp sequence that allows the port to identify itself as an HS port. This is useful for testing FS configurations with an HS host, hub, or device. 0 Allow the port to identify itself as High Speed. 1 Force the port to only connect at Full-speed. This bit is not defined in the EHCI specification. This bit is for debugging purposes.
PHCD	PHY Low Power Suspend. This bit is not defined in the EHCI specification. In host mode, the PHY can be put into low power suspend when the downstream device has been put into suspend mode or when no downstream device is connected. Low-power suspend is completely under the control of software. For the OTG module in device mode, the PHY can be put into low power suspend when the device is not running (USB_USBCMD Run/Stop = 0) or suspend signaling is detected on the USB. Low-power suspend is cleared automatically when the resume signaling has been detected or when forcing port resume. 0 Normal PHY operation. 1 Signal the PHY to enter low power suspend mode Reading this bit indicates the status of the PHY.
WKOC	Wake On Over-Current Enable. Writing this bit to 1 enables the port to be sensitive to over-current conditions as wake-up events. This field is 0 if Port Power (PP) is 0. This bit is (OTG/host mode only) for use by an external power control circuit.

Table 32-32. USB_PORTSC*n* field descriptions (continued)

Field	Description
WKDS	Wake On Disconnect Enable. Writing this bit to 1 enables the port to be sensitive to device disconnects as wake-up events. This field is 0 if Port Power (PP) is 0 or in device mode. This bit is (OTG/host mode only) for use by an external power control circuit.
WKCN	Wake On Connect Enable. Writing this bit to 1 enables the port to be sensitive to device connects as wake-up events. This field is 0 if Port Power (PP) is 0 or in device mode.
PTC	Port Test Control. Any value other than zero indicates the port is operating in test mode. 0000 Not Enabled. 0001 J_STATE. 0010 K_STATE. 0011 SE0_NAK. 0100 Packet. 0101 FORCE_ENABLE. 0110-1111 Reserved. Refer to Chapter 7 of the USB Specification Revision 2.0 [3] for details on each test mode.
PIC	Port Indicator Control. These bits control the link indicator signals. These signals are valid for host mode only. 00 Off. 01 Amber. 10 Green. 11 Undefined. Refer to the USB Specification Revision 2.0 [3] for a description on how these bits are to be used. This field is output from the module on the USB port control signals for use by an external LED driving circuit.
PO	Port Owner. This bit unconditionally goes to a 0 when the configured bit in the USB_CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 when the configured bit is zero. System software uses this field to release ownership of the port to a selected module (in the event that the attached device is not a high-speed device). Software writes a 1 to this bit when the attached device is not a high-speed device. A one in this bit means an internal companion controller owns and controls the port. Port owner hand-off is not implemented in this design, therefore this bit is always 0.
PP	Port Power. This bit represents the current setting of the switch (0 = off, 1 = on). When power is not available on a port (i.e., PP = 0), the port is non-functional and does not report attaches, detaches, etc. When an over-current condition is detected on a powered port, the PP bit in each affected port is transitioned by the host controller driver from a 1 to a 0 (removing power from the port). This feature is implemented in the host/OTG controller (PPC = 1). For the OTG module in a device-only implementation, port power control is not necessary. Therefore, PPC and PP equal 0.
LS	Line Status. These bits reflect the current logical levels of the USB D+ (bit 11) and D– (bit 10) signal lines. The use of line status by the host controller driver is not necessary (unlike EHCI) because hardware manages the connection of FS and LS. 00 SE0. 01 J-state. 10 K-state. 11 Undefined.
HSP	High Speed Port. This bit is redundant with the USB_PORTSC*n*[PSPD] bits. 1 Host/Device connected is operating in High Speed mode. 0 Host/Device connected is not operating in High Speed mode. This field is 0 if Port Power (PP) is 0.

Table 32-32. USB_PORTSC*n* field descriptions (continued)

Field	Description
PR	Port Reset. In host mode, when software writes a 1 to this bit, the bus-reset sequence as defined in the USB specification revision 2.0 is started. This bit automatically changes to 0 after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to 0 after the reset duration is timed in the driver. For the DR module in device mode, this bit is a read-only status bit. Device reset from the USB bus is also indicated in the USB_USBSTS register. 0 Port is not in Reset. 1 Port is in Reset. This field is 0 if Port Power (PP) is 0.
SUSP	Suspend In host mode: The port enabled bit (PE) and suspend (SUSP) bit define the port states as follows: 0x Disable. 10 Enable. 11 Suspend. When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Bit status does not change until the port is suspended and there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The module unconditionally sets this bit to 0 when software sets the Force Port Resume (FPR) bit to 0. The host controller ignores a write of 0 to this bit. If host software sets this bit to 1when the port is not enabled (the PE bit is 0), results are undefined. This field is 0 if Port Power (PP) is 0 in host mode. For the OTG module in device mode: 0 Port not in suspend state. Default. 1 Port in suspend state. In device mode, this bit is a read-only status bit.
FPR	Force Port Resume. This bit is not-EHCI compatible. 0 No resume (K-state) detected/driven on port. 1 Resume detected/driven on port. **In Host mode:** Software sets this bit to one to drive resume signaling. The controller sets this bit to one if a J-to-K transition is detected while the port is in the suspend state. When this bit transitions to 1, a J-to-K transition is detected, and the port change detect bit in the USB_USBSTS register is also set to one. This bit automatically changes to 0 after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to 0 after the resume duration is timed in the driver. When the controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (full-speed K) is driven on the port as long as this bit is set. This bit remains set until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller times the resume operation to clear the bit the port control state switches to HS or FS idle. This field is 0 if Port Power (PP) is 0 in host mode. **In Device mode:** After the device has been in suspend state for 5 msec or more, software must set this bit to one to drive resume signaling before clearing. the OTG controller sets this bit to one if a J-to-K transition is detected while the port is in the suspend state. The bit is cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition is detected, the port change detect bit in the USB_USBSTS register is also set to one.

Table 32-32. USB_PORTSC*n* field descriptions (continued)

Field	Description
OCC	Over-current Change. This bit gets set to one when there is a change to over-current active. Software clears this bit by writing a one to this bit position. For host/OTG implementations, you can provide over-current detection to the USB*n*_PWRFAULT signal for this condition. For device-only implementations, this bit must always be 0. 0 No over current. 1 Over current detect.
OCA	Over-current Active. This bit automatically transitions from one to zero when the over current condition is removed. For host/OTG implementations the user can provide over-current detection to the USB*n*_PWRFAULT signal for this condition. For device-only implementations this bit must always be 0. 0 Port not in over-current condition. 1 Port currently in over-current condition.
PEC	Port Enable/Disable Change. For the root hub, this bit is set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. In Device mode, the device port is always enabled. (This bit is zero). 0 No change. 1 Port disabled. This field is 0 if Port Power (PP) is 0.
PE	Port Enabled/Disabled. In host mode, ports can only be enabled by the controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. The bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host and bus events. When the port is disabled, (0) downstream propagation of data is blocked except for reset. This field is zero if port power(PP) is zero in host mode. In device mode (DR-only), the device port is always enabled. (This bit is one).
CSC	Connect Change Status. In host mode, this bit indicates a change has occurred in the port's current connect status. The controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, and hub hardware is setting an already-set bit (the bit remains set). Software clears this bit by writing a one to it. 0 No change. 1 Connect Status has changed. This field is 0 if Port Power (PP) is 0. In device mode, this bit is undefined.

Table 32-32. USB_PORTSC*n* field descriptions (continued)

Field	Description
CCS	Current Connect Status. In host mode: 0 No device present. 1 Device is present. This field is 0 if Port Power (PP) is 0 in host mode. In device mode: 0 Not attached. 1 Attached. A one indicates the device successfully attached and is operating in either high-speed or full-speed as indicated by the high speed port bit in this register. A zero indicates the device did not attach successfully or was forcibly disconnected by the software writing a zero to the run bit in the USB_USBCMD register. It does not state the device being disconnected or suspended.

32.2.4.18 On-The-Go Status and Control (USB_OTGSC) Register (Non-EHCI)

Both OTG modules implement one On-The-Go (OTG) Status and Control (USB_OTGSC) register. For more information, please refer to the *On-The-Go Supplement to the USB 2.0 Specification.*

The USB_OTGSC register has four sections:

- OTG Interrupt enables (Read/Write)
- OTG Interrupt status (Read/Write to Clear)
- OTG Status inputs (Read Only)
- OTG Controls (Read/Write)

The status inputs are de-bounced using a 1 ms time constant. Values on the status inputs that do not persist for more than 1 ms do not cause an update of the status inputs or cause and OTG interrupt.

Some of the reset values in this register are applicable only after the proper software driver has been loaded (e.g., for BSE, BSV, ASV, AVV, and ID).

This register is not defined in the EHCI specification.

Address: Base + 0x1A4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	DPIE	1msE	BSEIE	BSVIE	ASVIE	AVVIE	IDIE	0	DPIS	1msS	BSEIS	BSVIS	ASVIS	AVVIS	IDIS
W										w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	DPS	1msT	BSE	BSV	ASV	AVV	ID	HABA	HADP	IDPU	DP	OT	HAAR	VC	VD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-32. On-The-Go Status and Control (USB_OTGSC) Register

Table 32-33. USB_OTGSC field descriptions

Field	Description
DPIE	Data Pulse Interrupt Enable. 0 Disable. 1 Enable.
1msE	1 millisecond timer Interrupt Enable. 0 Disable. 1 Enable.
BSEIE	B Session End Interrupt Enable. 0 Disable. 1 Enable.
BSVIE	B Session Valid Interrupt Enable. 0 Disable. 1 Enable.
ASVIE	A Session Valid Interrupt Enable. 0 Disable. 1 Enable.
AVVIE	A VBus Valid Interrupt Enable. 0 Disable. 1 Enable.
IDIE	USB ID Interrupt Enable. 0 Disable. 1 Enable.
DPIS	Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USB_USBMODE[CM] equals Host (11) and PORTSC0[PP] equals Off (0). Software must write a 1 to clear this bit.
1msS	1 millisecond timer Interrupt Status. This bit is set once every millisecond. Software must write a 1 to clear this bit.
BSEIS	B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software must write a 1 to clear this bit.
BSVIS	B Session Valid Interrupt Status. This bit is set when VBus has risen above or fallen below the B session valid threshold (0.8 VDC). Software must write a 1 to clear this bit.
ASVIS	A Session Valid Interrupt Status. This bit is set when VBus has risen above or fallen below the A session valid threshold (0.8 VDC). Software must write a 1 to clear this bit.
AVVIS	A VBus Valid Interrupt Status. This bit is set when VBus has risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software must write a 1 to clear this bit.
IDIS	USB ID Interrupt Status. This bit is set when a change on the ID input is detected. Software must write a 1 to clear this bit.
DPS	Data Bus Pulsing Status. 0 No pulsing on port. 1 Pulsing detected on port.
1msT	1 millisecond timer toggle. This bit toggles once per millisecond.

Table 32-33. USB_OTGSC field descriptions (continued)

Field	Description
BSE	B Session End. 0 VBus is above the B session end threshold. 1 VBus is below the B session end threshold.
BSV	B Session Valid. 0 VBus is below the B session valid threshold. 1 VBus is above the B session valid threshold.
ASV	A Session Valid. 0 VBus is below the A session valid threshold. 1 VBus is above the A session valid threshold.
AVV	A VBus Valid. 0 VBus is below the A VBus valid threshold. 1 VBus is above the A VBus valid threshold.
ID	USB ID. 0 A device. 1 B device.
HABA	Hardware Assist B-Disconnect to A-connect. 0 Disabled. 1 Enable automatic B-disconnect to A-connect sequence.
HADP	Hardware Assist Data-Pulse. 0 1 Start Data Pulse Sequence.
IDPU	ID Pullup. This bit provide control over the ID pull-up resister: 0 Off 1 On When this bit is 0, the ID input is not sampled.
DP	Data Pulsing. 0 The pullup on DP is not asserted. 1 The pullup on DP is asserted for data pulsing during SRP.
OT	OTG Termination. This bit must be set when the OTG device is in device mode. 0 Disable pulldown on DM. 1 Enable pulldown on DM.
HAAR	Hardware Assist Auto-Reset. 0 Disabled. 1 Enable automatic reset after connect on host port.
VC	VBUS Charge. Setting this bit causes the VBus line to be charged. This is for VBus pulsing during SRP.
VD	VBUS discharge. Setting this bit causes VBus to discharge through a resistor.

32.2.4.19 USB Mode (USB_USBMODE) Register (Non-EHCI)

The USB Mode (USB_USBMODE) register controls the operating mode of the module. This register is not defined in the EHCI specification.

Address: Base + 0x1A8 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	VBPS	SDIS	SLOM	ES	CM	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-33. USB Mode (USB_USBMODE) Register

Table 32-34. USB_USBMODE field descriptions

Field	Description
VBPS	Vbus Power Select (0—Output is 0; 1—Output is one) This bit is connected to the vbus_pwr_select output and can be used for any generic control but is named to be used by logic that selects between an on-chip Vbus power source (charge pump) and an off-chip source in systems when both are available.
SDIS	Stream Disable. In host mode, setting this bit to a 1 ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB. Time duration to pre-fill the FIFO becomes significant when stream disable is active. See USB_TXFILLTUNING to characterize the adjustments needed for the scheduler when using this feature. In systems with high system bus utilization, setting this bit ensures no overruns or underruns during operation at the expense of link utilization. For those who desire optimal link performance, SDIS can be left clear, and the rules used under the description of the USB_TXFILLTUNING register to limit underruns/overruns. 0 Inactive. 1 Active. In device mode, setting this bit to a 1 disables double priming on both RX and TX for low bandwidth systems. This mode ensures that when the RX and TX buffers are sufficient to contain an entire packet, the standard double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. In high speed mode, all packets received are responded to with a NYET handshake when stream disable is active.
SLOM	Setup Lockout Mode. For the OTG module in device mode, this bit controls behavior of the setup lock mechanism. See Section 32.8.5.2, “Control Endpoint Operation Model.” 0 Setup Lockouts On. 1 Setup Lockouts Off (DCD requires use of Setup Data Buffer Tripwire in USB_USBCMD).
ES	Endian Select. This bit can change the byte ordering of the transfer buffers to match the host microprocessor bus architecture. The bit fields in the microprocessor interface and the DMA data structures (including the setup buffer within the device QH) are unaffected by the value of this bit, because they are based upon 32-bit words. 0 Little Endian—[Default]: first byte referenced in least significant byte of 32-bit word. 1 Big Endian—first byte referenced in most significant byte of 32-bit word.

Table 32-34. USB_USBMODE field descriptions (continued)

Field	Description
CM	Controller Mode. This register can only be written once after reset. If necessary to switch modes, software must reset the controller by writing to the RESET bit in the USB_USBCMD register before reprogramming this register. 00 Idle (Default for combination host/device). 01 Reserved. 10 Device Controller (Default for device only controller). 11 Host Controller (Default for host only controller). The OTG module defaults to the idle state and needs to be initialized to the desired operating mode after reset.

32.2.4.20 Endpoint Setup Status (USB_ENDPTSETUPSTAT) Register (Non-EHCI)

The Endpoint Setup Status Register (USB_ENDPTSETUPSTAT) contains the endpoint setup status. It is used only by the OTG module in device mode. This register is not defined in the EHCI specification.

Address: Base + 0x1AC Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	ENDPTSETUPSTAT			
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-34. Endpoint Setup Status Register (USB_ENDPTSETUPSTAT)

Table 32-35. USB_ENDPTSETUPSTAT field descriptions

Field	Description
ENDPTSETUPSTAT	Endpoint Setup Status. For every setup transaction received, a corresponding bit in this register is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to a respective bit after it has read the setup data from queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus timeouts while the setup lock our mechanism is engaged. See Section 32.8.4, “Managing Endpoints.” This register is used only in device mode.

32.2.4.21 Endpoint Initialization (USB_ENDPTPRIME) Register (Non-EHCI)

The Endpoint Initialization (USB_ENDPTPRIME) register is used to initialize endpoints. It is used only by the OTG module in device mode. This register is not defined in the EHCI specification.

Address: Base + 0x1B0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	PETB			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	PERB			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-35. Endpoint Initialization (USB_ENDPTPRIME) Register

Table 32-36. USB_ENDPTPRIME field descriptions

Field	Description
PETB	Prime Endpoint Transmit Buffer. For each endpoint, a corresponding bit requests a buffer to prepare for a transmit operation to respond to a USB IN/INTERRUPT transaction. Software should write a 1 to the corresponding bit when posting a new transfer descriptor to an endpoint. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **Note:** Hardware momentarily sets these bits during hardware re-priming operations when a dTD is retired, and the dQH is updated. PETB[3]—Endpoint #3 PETB[2]—Endpoint #2 PETB[1]—Endpoint #1 PETB[0]—Endpoint #0
PERB	Prime Endpoint Receive Buffer. For each endpoint, a corresponding bit requests a buffer to prepare for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a 1 to the corresponding bit when posting a new transfer descriptor to an endpoint. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **Note:** Hardware momentarily sets these bits during hardware re-priming operations when a dTD is retired, and the dQH is updated. PERB[3]—Endpoint #3 PERB[2]—Endpoint #2 PERB[1]—Endpoint #1 PERB[0]—Endpoint #0

32.2.4.22 Endpoint Flush (USB_ENDPTFLUSH) Register (Non-EHCI)

The Endpoint Flush (USB_ENDPTFLUSH) register is used only by the OTG module in device mode. This register is not defined in the EHCI specification.

Address: Base + 0x1B4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	FETB			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	FERB			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-36. Endpoint Flush (USB_ENDPTFLUSH) Register

Table 32-37. USB_ENDPTFLUSH field descriptions

Field	Description
FETB	Flush Endpoint Transmit Buffer. Writing a 1 to a bit(s) in this register causes the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. FETB[3]—Endpoint #3 FETB[2]—Endpoint #2 FETB[1]—Endpoint #1 FETB[0]—Endpoint #0
FERB	Flush Endpoint Receive Buffer. Writing a 1 to a bit(s) causes the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. FERB[3]—Endpoint #3 FERB[2]—Endpoint #2 FERB[1]—Endpoint #1 FERB[0]—Endpoint #0

32.2.4.23 Endpoint Status (USB_ENDPTSTATUS) Register

The Endpoint Status (USB_ENDPTSTATUS) register is not defined in the EHCI specification. This register is used only by the OTG module in device mode.

Address: Base + 0x1B8 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	ETBR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	ERBR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-37. Endpoint Status (USB_ENDPTSTATUS) Register

Table 32-38. USB_ENDPTSTATUS field descriptions

Field	Description
ETBR	Endpoint Transmit Buffer Ready. One bit for each endpoint indicates status of the respective endpoint buffer. Hardware sets this bit as a response to receiving a command from a corresponding bit in the USB_ENDPTPRIME register. There is always a delay between setting a bit in the USB_ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the USB_ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the USB_ENDPTFLUSH register. **Note:** Hardware momentarily clears these bits during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ETBR[3]—Endpoint #3 ETBR[2]—Endpoint #2 ETBR[1]—Endpoint #1 ETBR[0]—Endpoint #0
ERBR	Endpoint Receive Buffer Ready. One bit for each endpoint indicates status of the respective endpoint buffer. Hardware sets this bit to a one by the hardware as a response to receiving a command from a corresponding bit in the USB_ENDPTPRIME register. There is always a delay between setting a bit in the USB_ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the USB_ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the USB_ENDPTFLUSH register. **Note:** Hardware momentarily clears these bits during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ERBR[3]—Endpoint #3 ERBR[2]—Endpoint #2 ERBR[1]—Endpoint #1 ERBR[0]—Endpoint #0

32.2.4.24 Endpoint Complete (USB_ENDPTCOMPLETE) Register (Non-EHCI)

The Endpoint Complete (USB_ENDPTCOMPLETE) register is not defined in the EHCI specification. This register is used only by the OTG module in device mode.

Address: Base + 0x1BC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	ETCE			
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	ERCE			
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-38. Endpoint Complete Register (USB_ENDPTCOMPLETE)

Table 32-39. USB_ENDPTCOMPLETE field descriptions

Field	Description
ETCE	Endpoint Transmit Complete Event. Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the transfer descriptor, this bit is set simultaneously with the USBINT. Writing a 1 clears the corresponding bit in this register. ETCE[3]—Endpoint #3 ETCE[2]—Endpoint #2 ETCE[1]—Endpoint #1 ETCE[0]—Endpoint #0
ERCE	Endpoint Receive Complete Event. Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the transfer descriptor, this bit is set simultaneously with the USBINT. Writing a 1 clears the corresponding bit in this register. ERCE[3]—Endpoint #3 ERCE[2]—Endpoint #2 ERCE[1]—Endpoint #1 ERCE[0]—Endpoint #0

32.2.4.25 Endpoint Control Register 0 (USB_ENDPTCTRL0)

Every device implements endpoint 0 as a control endpoint. This register is not defined in the EHCI specification.

Address: Base + 0x1C0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TXE	0	0	0	TXT		0	TXS
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RXE	0	0	0	RXT		0	RXS
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 32-39. Endpoint Control Register 0 (USB_ENDPTCTRL0)

Table 32-40. USB_ENDPTCTRL0 field descriptions

Field	Description
TXE	TX Endpoint Enable. Endpoint zero is always enabled. 1 Enable.
TXT	TX Endpoint type. Endpoint zero is always a control endpoint (00).
TXS	TX Endpoint Stall. Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the host. It continues returning STALL until the software clears this bit or it is automatically cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit continues to be cleared by hardware until the associated bit in the USB_ENDPTSETUPSTAT register is cleared. **Note:** There is a slight delay (50 clocks max.) between the associated bit in the USB_ENDPTSETUPSTAT register being cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software observes this delay. However, should the DCD observe that the stall bit is not set after writing a 1 to it, follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated bit in the USB_ENDPTSETUPSTAT register. 1 Endpoint Stalled. 0 Endpoint OK.
RXE	RX Endpoint Enable. Endpoint zero is always enabled. 1 Enabled.
RXT	RX Endpoint type. Endpoint zero is always a control endpoint (00).
RXS	RX Endpoint Stall. Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the host. It continues returning STALL until software clears this bit or it is automatically cleared upon receipt of a new SETUP request. After receiving a SETUP request, hardware continues to clear this bit until the associated bit in the USB_ENDPTSETUPSTAT register is cleared. **Note:** There is a slight delay (50 clocks max.) between the associated bit in the USB_ENDPTSETUPSTAT register being cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software observes this delay. However, should the DCD observe that the stall bit is not set after writing a 1 to it, follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated bit in the USB_ENDPTSETUPSTAT register. 0 Endpoint OK. 1 Endpoint Stalled.

32.2.4.26 Endpoint Control Register *n* (USB_ENDPTCTRL*n*) (Non-EHCI)

There is an USB_ENDPTCTRL*n* register of each endpoint in a device. USB_ENDPTCTRL0 is used as a control endpoint. The remaining endpoints are not used as control endpoints.

These registers are not defined in the EHCI specification.

Address: Base + 0x1C4 (USB_ENDPTCTRL1)
Base + 0x1C8 (USB_ENDPTCTRL2)
Base + 0x1CC (USB_ENDPTCTRL3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TXE	TXR	TXI	0	TXT		TXD	TXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RXE	RXR	RXI	0	RXT		RXD	RXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-40. Endpoint Control Register *n* (USB_ENDPTCTRL*n*)

Table 32-41. USB_ENDPTCTRL*n* field descriptions

Field	Description
TXE	TX Endpoint Enable. 0 Disabled. 1 Enabled.
TXR	TX Data Toggle Reset. When a configuration event is received for this endpoint, software must write a one to this bit to synchronize the data PIDs between the host and device.
TXI	TX Data Toggle Inhibit. This bit is used only for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 PID Sequencing Enabled. 1 PID Sequencing Disabled.
TXT	TX Endpoint type. 00 Control. 01 Isochronous. 10 Bulk. 11 Interrupt.
TXD	TX Endpoint Data Source. This bit should always be written as 0, which selects the Dual Port Memory/DMA Engine as the source.
TXS	TX Endpoint Stall. This bit is set automatically upon receipt of a setup request if this endpoint is not configured as a control endpoint. It is cleared automatically upon receipt of a setup request if this endpoint is configured as a control endpoint. Software can write a 1 to this bit to force the endpoint to return a stall handshake to the host. It continues returning stall until this bit is cleared by software or automatically cleared as above. Software can write a 1 to this bit to force the endpoint to return a stall handshake to the Host. This control continues to stall until this bit is either cleared by software or automatically cleared as above for control endpoints. **Note:** [Control endpoint types only] There is a slight delay (50 clocks max.) between the associated bit in the USB_ENDPTSETUPSTAT register being cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software observes this delay. However, should the DCD observe that the stall bit is not set after writing a 1 to it, follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated bit in the USB_ENDPTSETUPSTAT register. 0 Endpoint OK. 1 Endpoint Stalled.

Table 32-41. USB_ENDPTCTRL*n* field descriptions (continued)

Field	Description
RXE	RX Endpoint Enable. 0 Disabled. 1 Enabled.
RXR	RX Data Toggle Reset. When a configuration event is received for this endpoint, software must write a 1 to this bit to synchronize the data PIDs between the host and device.
RXI	RX Data Toggle Inhibit. This bit is used only for test and should always be written as 0. Writing a 1 to this bit causes this endpoint to ignore the data toggle sequence and always accepts data packets regardless of their data PID. 0 PID Sequencing Disabled. 1 PID Sequencing Enabled.
RXT	RX Endpoint type. 00 Control. 01 Isochronous. 10 Bulk. 11 Interrupt.
RXD	RX Endpoint Data Sink. This bit should always be written as 0, which selects the dual port memory/DMA engine as the sink.
RXS	RX Endpoint Stall. This bit is set automatically upon receipt of a SETUP request if this endpoint is not configured as a control endpoint. It is cleared automatically upon receipt of a SETUP request if this endpoint is configured as a control endpoint, Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the host. It continues returning STALL until this bit is cleared by software or automatically cleared as above. Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the host. This control continues to STALL until this bit is cleared by software or automatically cleared as above for control endpoints. **Note:** [Control endpoint types only] There is a slight delay (50 clocks max.) between the associated bit in the USB_ENDPTSETUPSTAT register being cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software observeS this delay. However, should the DCD observe that the stall bit is not set after writing a 1 to it, follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated bit in the USB_ENDPTSETUPSTAT register. 0 Endpoint OK. 1 Endpoint Stalled.

32.2.4.27 USB General Control (USB_USBGENCTRL) Register (Non-EHCI)

The USB General Control (USB_USBGENCTRL) register is shown in Figure 32-41. This register uses big endian byte ordering. This register is not defined in the EHCI specification.

Address: Base + 0x200 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	WU_INT_CLR	ULPI_SEL	PPP	PFP	WU_ULPI_EN	WU_IE
W											w1c					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-41. USB General Control (USB_USBGENCTRL) Register

Table 32-42. USB_USBGENCTRL field descriptions

Field	Description
WU_INT_CLR	Wake-up Interrupt clear. 0 Default, no action. 1 Clear the wake-up interrupt.
ULPI_SEL	ULPI I/F Select Not used in this implementation.
PPP	Port Power Polarity. Not used in this implementation.
PFP	Power Fault Polarity. Not used in this implementation.
WU_ULPI_EN	Wakeup on ULPI Interrupt Event. This bit is used to enable the wake up from the ULPI I/F. 0 Wake Up Interrupt Disabled. 1 Wake Up Interrupt Enabled.
WU_IE	WAKEUP INTERRUPT ENABLE. This bit is used to enable the low power wakeup interrupt. 0 Low power wakeup interrupt disabled. 1 Low power wakeup interrupt enabled.

32.3 Functional Description

Both modules, USB1/USB2, can be broken down into functional sub-blocks described below.

32.3.1 System Interface

The system interface block contains all the control and status registers that allow the CPU core to interface to the module. These registers allow the processor to control the configuration of the module, ascertain the capabilities of each module and control the module's operation.

32.3.2 DMA Engine

Both USB controllers contain local DMA engines. The DMA Engine interface is responsible for moving all of the data to transfer over the USB between the controller and buffers in system memory. Like the system interface block, the DMA engine block uses a simple synchronous bus signaling protocol.

The DMA controllers must access both control information and packet data from system memory. The control information is contained in link-list based queue structures. The DMA controllers have state machines able to parse data structures defined in the EHCI specification. In host mode, the data structures are EHCI compliant and represent queues of transfers performed by the host controller. In device mode, the data structures are similar to those in the EHCI specification and allow device responses to be queued for each of the active pipes in the device.

32.3.3 FIFO RAM Controller

The FIFO RAM controller is for context information and to control FIFOs between the protocol engine and the DMA controller. These FIFOs decouple the system processor/memory bus requests from the extremely tight timing required by USB.

The use of the FIFO buffers differs between host and device mode operation. In host mode, a single data channel is maintained in each direction through the buffer memory. In device mode, multiple FIFO channels are maintained for each of the active endpoints in the system.

For both modules (USB1 and USB2) device operation uses a single 256-byte RX buffer and a 512-byte TX buffer for each endpoint. The 512-byte buffers allow the modules to buffer a complete HS bulk packet.

The USB1 and USB2 module interfaces to ULPI compatible PHY. The primary function of the port controller block is to isolate the rest of the module from the transceiver and to move all of the transceiver signaling into the primary clock domain of the module. This allows the module to run synchronously with the system processor and it's associated resources.

32.4 OTG Operations

32.4.1 Register Bits

In the previous section, the register interface has behaviors described for device and host mode. However, during OTG operations, it is necessary to perform tasks independent of the controller mode.

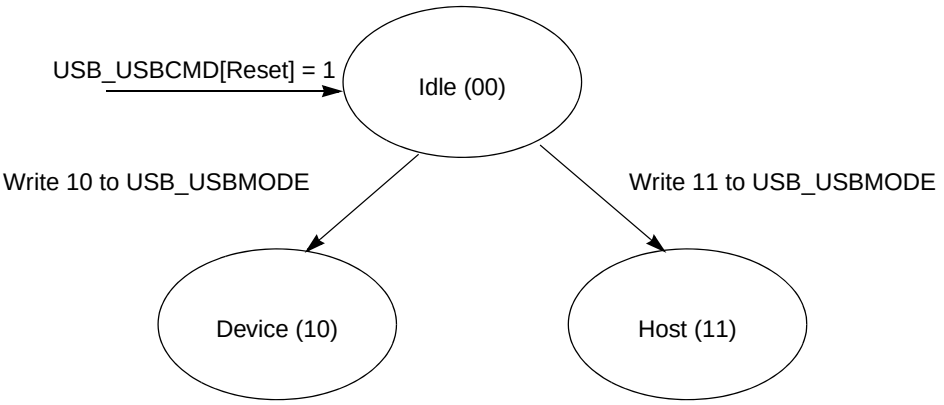


Figure 32-42. Controller Mode

Figure 32-42 also shows that the only way to transition the controller mode out of host or device mode is with the controller reset bit. Therefore, it is also necessary for the OTG tasks to be performed independent of a controller reset as well as independent of the controller mode.

To this end, the following list contains the register bits used for OTG operations, which are independent of the controller mode and are also not affected by a write to the reset bit in the USB_USBCMD register.

- All Identification Registers
- All Device/Host Capability Registers
- USB_OTGSC: All bits
- USB_PORTSC*n*:
  - Physical Interface Select
  - Physical Interface Serial Select
  - Physical Interface Data Width
  - Physical Interface Low Power
  - Physical Interface Wake Signals
  - Port Indicators
  - Port Power

32.4.2 Hardware Assist

The hardware assist provides automated response and sequencing that may not be possible to software with significant interrupt latency response times. The use of this additional circuitry is optional and can be used to assist the three sequences below.

32.4.2.1 Auto-Reset

When the HAAR is set to one, the host automatically starts a reset after a connect event. This shortcuts the normal process where software is notified of the connect event and starts the reset. Software receives notification of the connect event, but should not write the reset bit when the HAAR is set. Software is

notified again after the reset is complete via the enable change bit in the USB_PORTSC*n* register that causes a port change interrupt.

This assist ensures the OTG parameter TB_ACON_BSE0_MAX = 1 ms is met.

32.4.2.2 Data-Pulse

Writing a one to HADP starts a data pulse of approximately 7ms in duration and then automatically cease the data pulsing. During the data pulse, the DP is set and then cleared. This automation relieves software from accurately controlling the data-pulse duration. During the data pulse, the HCD can poll to see that the HADP and DP bit have returned low to recognize the completion or simply launch the data pulse and wait to see if a VBUS Valid interrupt occurs when the A-side supplies bus power.

This assist ensures data pulsing meets the OTG requirement of > 5 ms and < 10 ms.

32.4.2.3 B-Disconnect to A-Connect

During HNP, the B-disconnect occurs from the OTG A_suspend state and within 3 ms, the A-device must enable the pullup on the DP leg in the A-peripheral state. When HABA is set, the Host Controller port is in suspend mode, and the device disconnects, then this hardware assist begins.

1. Reset the OTG core.
2. Write the OTG core into device mode.
3. Write a 1 to the device run bit and enable necessary interrupts including:
   a) USB Reset Enable (URE); enables interrupt on USB bus reset to device.
   b) Sleep Enable (SLE); enables interrupt on device suspend.
   c) Port Change Detect Enable (PCE); enables interrupt on device connect.

When software has enabled this hardware assist, it must not interfere during the transition and should not write any register in the core until it gets an interrupt from the device controller signifying that a reset interrupt has occurred or at least first verify that the core has entered device mode. HCD/DCD must not activate the core soft reset at any time since this action is performed by hardware. During the transition, the software may see an interrupt from the disconnect and/or other spurious interrupts (i.e., SOF/etc.) that may or may not cascade and my be cleared by the soft reset depending on the software response time.

After the core has entered device mode by the hardware assist, the DCD must ensure that the ENDPTLISTADDR is programmed properly before the host sends a setup packet. Since the end of the reset duration, which may be initiated quickly (a few microseconds) after connect, requires at a minimum 50 ms, this is the time for which the DCD must be ready to accept setup packets after having received notification that the reset has been detected or simply that the OTG is in device mode which ever occurs first.

In the case where the A-peripheral fails to see a reset after the controller enters device mode and engages the DP-pullup, the device controller interrupt the DCD signifying that a suspend has occurred.

This assist ensures the parameter TA_BDIS_ACON_MAX = 3 ms is met.

32.5 Host Data Structures

This section defines the interface data structures used to communicate control, status, and data between HCD (software) and the enhanced host controller (hardware). The data structure definitions in this section support a 32-bit memory buffer address space. The interface consists of a periodic schedule, periodic frame list, asynchronous schedule, isochronous transaction descriptors, split-transaction isochronous transfer descriptors, queue heads, and queue element transfer descriptors.

The periodic frame list is the root of all periodic (isochronous and interrupt transfer type) support for the host controller interface. The asynchronous list is the root for all the bulk and control transfer type support. Isochronous data streams are managed using isochronous transaction descriptors. Isochronous split-transaction data streams are managed with split-transaction isochronous transfer descriptors. All interrupt, control, and bulk data streams are managed via queue heads and queue element transfer descriptors. These data structures are optimized to reduce the total memory footprint of the schedule and to reduce (on average) the number of memory accesses needed to execute a USB transaction.

Software must ensure that no interface data structure reachable by the EHCI host controller spans a 4K-page boundary.

The data structures defined in this section are (from the host controller's perspective) a mix of read-only and read/writable fields. The host controller must preserve the read-only fields on all data structure writes.

32.5.1 Periodic Frame List

Figure 32-43 shows the organization of the periodic schedule. This schedule is for all periodic transfers (isochronous and interrupt). The periodic schedule is referenced from the operational registers space using the USB_PERIODICLISTBASE address register and the USB_FRINDEX register. The periodic schedule is based on an array of pointers called the Periodic Frame List. The USB_PERIODICLISTBASE address register is combined with the USB_FRINDEX register to produce a memory pointer into the frame list. The periodic frame list implements a sliding window of work over time.

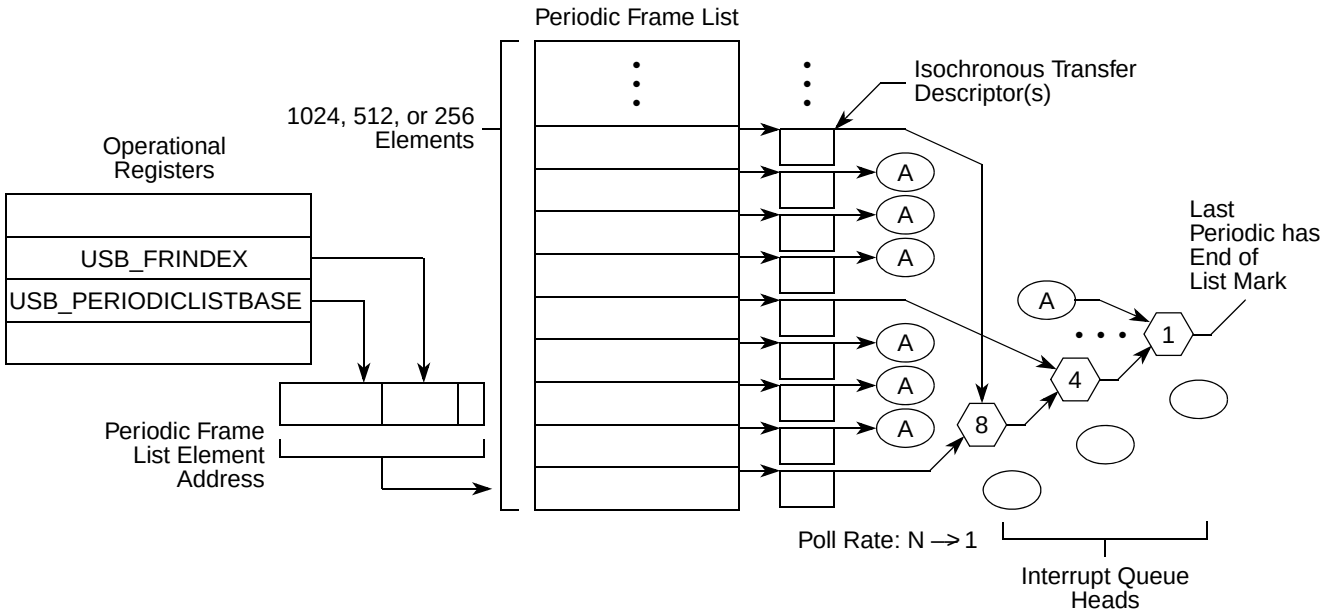


Figure 32-43. Periodic Schedule Organization

Split transaction interrupt, bulk and control are also managed using queue heads and queue element transfer descriptors.

The periodic frame list is a 4K-page aligned array of frame list link pointers. The length of the frame list may be programmable. The programmability of the periodic frame list is exported to system software via the USB_HCCPARAMS register. If non-programmable, the length is 1024 elements. If programmable, system software can select the length as one of 256, 512, or 1024 elements. An implementation must support all three sizes. Programming the size (that is, the number of elements) is accomplished by system software writing the appropriate value into frame list size field in the USB_USBCMD register.

Frame list link pointers direct the host controller to the first work item in the frame's periodic schedule for the current micro-frame. The link pointers are aligned on doubleword boundaries within the frame list. Figure 32-44 shows the format for the frame list link pointer.

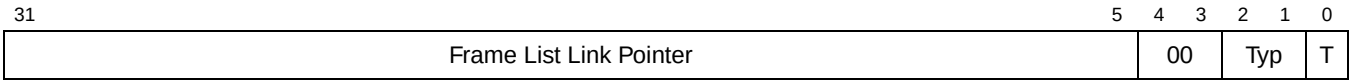


Figure 32-44. Frame List Link Pointer Format

Frame list link pointers always reference memory objects that are 32-byte aligned. The referenced object may be an isochronous transfer descriptor for high-speed devices, a split-transaction isochronous transfer descriptor (for full-speed isochronous endpoints), or a queue head (used to support high-, full- and low-speed interrupt). System software should not place non-periodic schedule items into the periodic schedule. The least significant bits in a frame list pointer key the host controller as to the type of object the pointer is referencing.

The least significant bit is the T-Bit (bit 0). When this bit is set, the host controller never uses the value of the frame list pointer as a physical memory pointer. The Typ field is used to indicate the exact type of data structure referenced by this pointer. The value encodings for the Typ field are given in Table 32-43.

Table 32-43. Typ Field Encodings

Typ	Description
00	Isochronous Transfer Descriptor
01	Queue Head
10	Split Transaction Isochronous Transfer Descriptor
11	Frame Span Traversal Node.

32.5.2 Asynchronous List Queue Head Pointer

The asynchronous transfer list (based at the USB_ASYNCLISTADDR register) is where all the control and bulk transfers are managed. Host controllers only use this list when it reaches the end of the periodic list, the periodic list is disabled, or the periodic list is empty.

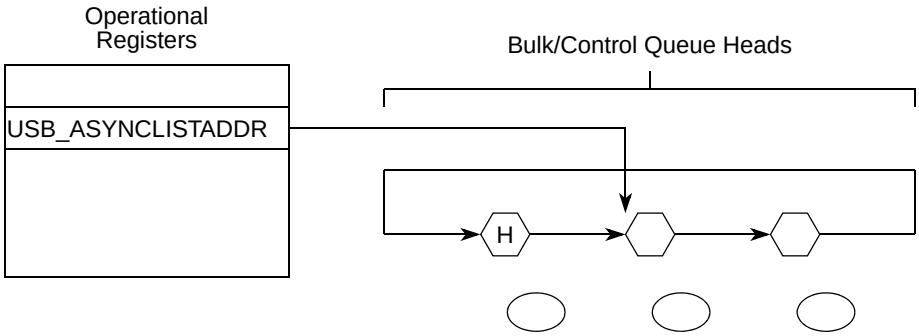


Figure 32-45. Asynchronous Schedule Organization

The asynchronous list is a simple circular list of queue heads. The USB_ASYNCLISTADDR register is a pointer to the next queue head. This implements a pure round-robin service for all queue heads linked into the asynchronous list.

32.5.3 Isochronous (High-Speed) Transfer Descriptor (iTD)

The format of an isochronous transfer descriptor is illustrated in Figure 32-46. This structure is used only for high-speed isochronous endpoints. All other transfer types should use queue structures. Isochronous TDs must be aligned on a 32-byte boundary.

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	offset
Next Link Pointer		00 Typ T	0x00
Status[1] Transaction 0 Length[1]	IOC	PG[2] Transaction 0 Offset[2]	0x04
Status[1] Transaction 1 Length[1]	IOC	PG[2] Transaction 1 Offset[2]	0x08
Status[1] Transaction 2 Length[1]	IOC	PG[2] Transaction 2 Offset[2]	0x0C
Status[1] Transaction 3 Length[1]	IOC	PG[2] Transaction 3 Offset[2]	0x10
Status[1] Transaction 4 Length[1]	IOC	PG[2] Transaction 4 Offset[2]	0x14
Status[1] Transaction 5 Length[1]	IOC	PG[2] Transaction 5 Offset[2]	0x18
Status[1] Transaction 6 Length[1]	IOC	PG[2] Transaction 6 Offset[2]	0x1C
Status[1] Transaction 7 Length[1]	IOC	PG[2] Transaction 7 Offset[2]	0x20
Buffer Pointer (Page 0)		EndPt R Device Address	0x24
Buffer Pointer (Page 1)		I/O Maximum Packet Size	0x28
Buffer Pointer (Page 2)		Reserved Mult	0x2C
Buffer Pointer (Page 3)		Reserved	0x30
Buffer Pointer (Page 4)		Reserved	0x34
Buffer Pointer (Page 5)		Reserved	0x38
Buffer Pointer (Page 6)		Reserved	0x3C

Figure 32-46. Isochronous Transaction Descriptor (iTD)

1 Host controller read/write; all others read-only.

2 These fields may be modified by the host controller if the I/O field indicates an OUT.

32.5.3.1 Next Link Pointer

The first doubleword of an iTD is a pointer to the next schedule data structure.

Table 32-44. Next Schedule Element Pointer

Field	Description
31–5 Link Pointer	These bits correspond to memory address signals [31:5], respectively. This field points to another isochronous transaction descriptor (iTD/siTD) or queue head (QH).
4,3	Reserved. These bits are reserved and their value has no effect on operation. Software should initialize this field to 0.
2,1 Typ	This field indicates to the host controller whether the item referenced is an iTD, siTD, or a QH. This allows the host controller to perform the proper type of processing on the item after it is fetched. Value encodings are: 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor 11 FSTN (frame span traversal node)
0 T	Terminate 1 Link Pointer field is not valid. 0 Link Pointer field is valid.

32.5.3.2 iTD Transaction Status and Control List

doublewords one through eight are eight slots of transaction control and status. Each transaction description includes:

- Status results field
- Transaction length (bytes to send for OUT transactions and bytes received for IN transactions).
- Buffer offset. The PG and transaction *n* offset fields are used with the buffer pointer list to construct the starting buffer address for the transaction.

The host controller uses the information in each transaction description plus the endpoint information contained in the first three doublewords of the buffer page pointer list to execute a transaction on the USB.

Table 32-45. iTD Transaction Status and Control

Field	Description
31–28 Status	This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding: 31 Active. Set by software to enable the execution of an isochronous transaction by the host controller. When the transaction associated with this descriptor is completed, the host controller sets this bit to 0 indicating that a transaction for this element should not be executed when it is next encountered in the schedule. 30 Data Buffer Error. Set by the host controller during status update to indicate that the host controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (underrun). If an overrun condition occurs, no action is necessary. 29 Babble Detected. Set by the host controller during status update when babble is detected during the transaction generated by this descriptor. 28 Transaction Error (XactErr). Set by the host controller during status update in the case where the host did not receive a valid response from the device (Time-out, CRC, Bad PID, etc.). This bit may only be set for isochronous IN transactions.
27–16 Transaction *n* Length	For an OUT, this field is the number of data bytes the host controller sends during the transaction. The host controller is not required to update this field to reflect the actual number of bytes transferred during the transfer. For an IN, the initial value of the endpoint to deliver. During the status update, the host controller writes back this field with the number of bytes the host expects to receive. The value in this register is the actual byte count (for example, 0 zero length data, 1 one byte, 2 two bytes, etc.). The maximum value this field may contain is 0xC00 (3072).
15 IOC	Interrupt on complete. If this bit is set, it specifies that when this transaction completes, the host controller should issue an interrupt at the next interrupt threshold.
14–12 PG	These bits are set by software to indicate which of the buffer page pointers the offset field in this slot should be concatenated to produce the starting memory address for this transaction. The valid range of values for this field is 0 to 6.
11–0 Transaction *n* Offset	This field is a value that is an offset, expressed in bytes, from the beginning of a buffer. This field is concatenated onto the buffer page pointer indicated in the adjacent PG field to produce the starting buffer address for this transaction.

32.5.3.3 iTD Buffer Page Pointer List (Plus)

Doublewords 9-15 of an isochronous transaction descriptor are nominally page pointers (4K aligned) to the data buffer for this transfer descriptor. This data structure requires the associated data buffer to be contiguous (relative to virtual memory), but allows the physical memory pages to be non-contiguous. Seven page pointers are provided to support the expression of eight isochronous transfers. The seven

pointers allow for 3 (transactions) × 1024 (maximum packet size × 8 (transaction records) = 24,576 bytes to be moved with this data structure, regardless of the alignment offset of the first page.

Because each pointer is a 4K aligned page pointer, the least significant 12 bits in several of the page pointers are for other purposes.

Table 32-46. Buffer Pointer Page 0 (Plus)

Field	Description
31:12 Buffer Pointer (Page 0)	This is a 4 K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11:8 EndPt	This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Reserved. Reserved for future use and should be initialized by software to 0.
6:0 Device Address	This field selects the specific device serving as the data source or sink.

Table 32-47. iTD Buffer Pointer Page 1 (Plus)

Field	Description
31:12 Buffer Pointer (Page 1)	This is a 4 K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11 I/O	Direction (I/O). This field encodes whether the high-speed transaction should use an IN or OUT PID. 0 OUT 1 IN
10:0 Maximum Packet Size	This directly corresponds to the maximum packet size of the associated endpoint (*wMaxPacketSize*). This field is used for high-bandwidth endpoints where more than one transaction is issued per transaction description (.for example, per micro-frame). This field is used with the Multi field to support high-bandwidth pipes. This field is also used for all IN transfers to detect packet babble. Software should not set a value larger than 1024 (400h). Any larger value yields undefined results.

Table 32-48. Buffer Pointer Page 2 (Plus)

Field	Description
31:12 Buffer Pointer (Page 2)	This is a 4 K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11:2	Reserved. This bit reserved for future use and should be cleared.
1:0 Mult	This field is used to indicate to the host controller the number of transactions that should be executed per transaction description (for example, per micro-frame). The valid values are: 00 Reserved. A 0 in this field yields undefined results. 01 One transaction to be issued for this endpoint per micro-frame 10 Two transactions to be issued for this endpoint per micro-frame 11 Three transactions to be issued for this endpoint per micro-frame

Table 32-49. Buffer Pointer Page 3-6

Field	Description
31:12 Buffer Pointer	This is a 4 K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11:2	Reserved. These bits reserved for future use and should be cleared.

32.5.4 Split Transaction Isochronous Transfer Descriptor (siTD)

All full-speed isochronous transfers through the internal transaction translator are managed using the siTD data structure. This data structure satisfies the operational requirements for managing the split transaction protocol.

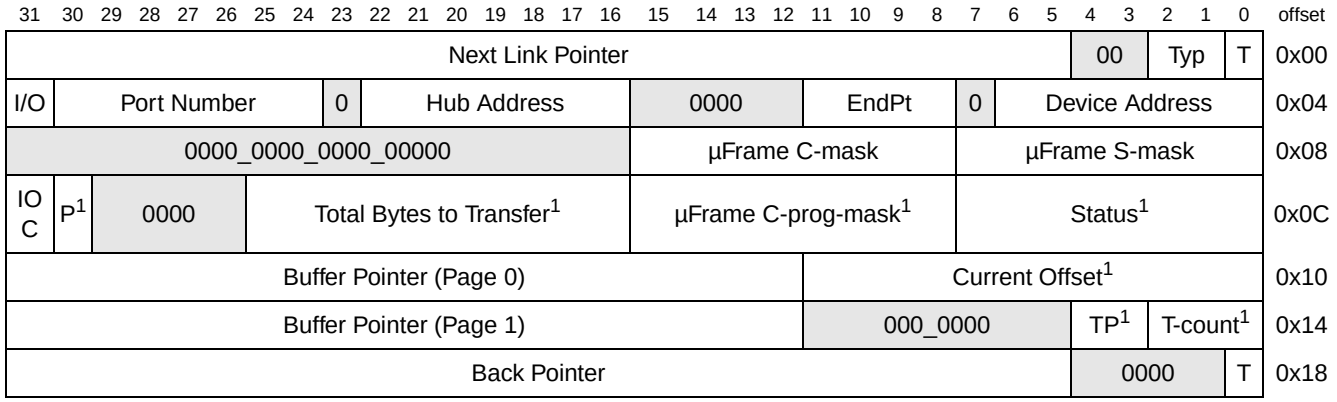


Figure 32-47. Split-Transaction Isochronous Transaction Descriptor (siTD)

1 Host controller read/write; all others read-only.

32.5.4.1 Next Link Pointer

Doubleword0 of an siTD is a pointer to the next schedule data structure.

Table 32-50. Next Link Pointer

Bit	Description
31:5 Next Link Pointer	This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4:3	Reserved. These bits must be written as zeros.
2:1 Typ	This field indicates to the host controller whether the item referenced is an iTD/siTD or a QH. This allows the host controller to perform the proper type of processing on the item after it is fetched. Value encodings are: 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor 11 FSTN (frame span traversal node)
0 T	Terminate. 0 Link Pointer is valid. 1 Link Pointer field is not valid.

32.5.4.2 siTD Endpoint Capabilities/Characteristics

Doublewords 1 and 2 specify static information about the full-speed endpoint, the addressing of the parent companion controller, and micro-frame scheduling control.

Table 32-51. Endpoint and Transaction Translator Characteristics

Bit	Description
31 I/O	Direction (I/O). This field encodes whether the full-speed transaction should be an IN or OUT. 0 OUT 1 IN
30:24 Port Number	This field is the port number of the recipient transaction translator.
23	Reserved. Bit reserved and should be cleared.
22:16 Hub Address	This field holds the device address of the companion controller's hub.
15:12	Reserved. Field reserved and should be cleared.
11:8 EndPt	Endpoint Number. This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Reserved. Bit is reserved for future use. It should be cleared.
6:0 Device Address	This field selects the specific device serving as the data source or sink.

Table 32-52. Micro-frame Schedule Control

Bit	Description
31:16	Reserved. This field reserved for future use. It should be cleared.
15:8 µFrame C-mask	Split Completion Mask. This field (along with the active and SplitX- state fields in the status byte) is used to determine during which micro-frames the host controller should execute complete-split transactions. When the criteria for using this field is met, an all zeros value has undefined behavior. The host controller uses the value of the three low-order bits of the USB_FRINDEX register to index into this bit field. If the USB_FRINDEX register value indexes to a position where the µFrame C-Mask field is a one, then this siTD is a candidate for transaction execution. There may be more than one bit in this mask set.
7:0 µFrame S-mask	Split Start Mask. This field (along with the active and SplitX-state fields in the status byte) is used to determine during which micro-frames the host controller should execute start-split transactions. The host controller uses the value of the three low-order bits of the USB_FRINDEX register to index into this bit field. If the USB_FRINDEX register value indexes to a position where the µFrame S-mask field is a one, then this siTD is a candidate for transaction execution. An all zeros value in this field, in combination with existing periodic frame list has undefined results.

32.5.4.3 siTD Transfer State

Doublewords 3-6 are used to manage the state of the transfer.

Table 32-53. siTD Transfer Status and Control

<table>
<tr><th>Bit</th><th colspan="2">Description</th></tr>
<tr><td>31
IOC</td><td colspan="2">Interrupt On Complete
0 Do not interrupt when transaction is complete.
1 Do interrupt when transaction is complete. When the host controller determines that the split transaction has completed, it asserts a hardware interrupt at the next interrupt threshold.</td></tr>
<tr><td>30
P</td><td colspan="2">Page Select. Used to indicate which data page pointer should be concatenated with the CurrentOffset field to construct a data buffer pointer
0 selects Page 0 pointer
1 selects Page 1 pointer
The host controller is not required to write this field back when the siTD is retired (Active bit transitioned from a 1 to a 0).</td></tr>
<tr><td>29:26</td><td colspan="2">Reserved. This field reserved for future use and should be cleared.</td></tr>
<tr><td>25:16
Total Bytes to Transfer</td><td colspan="2">Software initializes this field to the total number of bytes expected in this transfer. Maximum value is 1023 (3FFh)</td></tr>
<tr><td>15:8
µFrame C-prog-mask</td><td colspan="2">Split complete progress mask. The host controller uses this field to record which split-completes have been executed.</td></tr>
<tr><td rowspan="10">7:0
Status</td><td colspan="2">This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:</td></tr>
<tr><td>Status Bit</td><td>Definition</td></tr>
<tr><td>7</td><td>Active. Set by software to enable the execution of an isochronous split transaction by the host controller.</td></tr>
<tr><td>6</td><td>ERR. Set by the host controller when an ERR response is received from the Companion Controller.</td></tr>
<tr><td>5</td><td>Data Buffer Error. Set by the host controller during status update to indicate the host controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). In the case of an under run, the host controller transmits an incorrect CRC (thus invalidating the data at the endpoint). If an overrun condition occurs, no action is necessary.</td></tr>
<tr><td>4</td><td>Babble Detected. Set by the host controller during status update when babble is detected during the transaction generated by this descriptor.</td></tr>
<tr><td>3</td><td>Transaction Error (XactErr). Set by the host controller during status update in the case where the host did not receive a valid response from the device (Time-out, CRC, Bad PID, etc.). This bit is only set for IN transactions.</td></tr>
<tr><td>2</td><td>Missed Micro-Frame. The host controller detected that a host-induced hold- off caused the host controller to miss a required complete-split transaction.</td></tr>
<tr><td>1</td><td>Split Transaction State (SplitXstate). The bit encodings are:
0 Do Start Split. This value directs the host controller to issue a start split transaction to the endpoint when a match is encountered in the S-mask.
1 Do Complete Split. This value directs the host controller to issue a complete split transaction to the endpoint when a match is encountered in the C-mask.</td></tr>
<tr><td>0</td><td>Reserved. Bit reserved for future use and should be cleared.</td></tr>
</table>

32.5.4.4 siTD Buffer Pointer List (Plus)

Doublewords 4 and 5 are the data buffer page pointers for the transfer. This structure supports one physical page cross. The most significant 20 bits of each doubleword in this section are the 4 KB (page) aligned buffer pointers. The least significant 12 bits of each doubleword is an additional transfer state.

Table 32-54. siTD Buffer Pointer Page 0 (Plus)

Bit	Description
31:12 Buffer Pointer (Page 0)	Bits [31:12] is a 4K page-aligned, physical memory address. These bits correspond to physical address bits [31:12] respectively. The field P specifies the current active pointer
11:0 Current Offset	The 12 least significant bits of the Page 0 pointer is the current byte offset for the current page pointer (as selected with the page indicator bit (P field)). The host controller is not required to write this field back when the siTD is retired (Active bit transitioned from a one to a zero).

Table 32-55. siTD Buffer Pointer Page 1 (Plus)

Bit	Description
31:12 Buffer Pointer (Page 1)	Bits [31:12] is a 4 KB page-aligned, physical memory address. These bits correspond to physical address bits [31:12] respectively. The field P specifies the current active pointer
11:5	Reserved.
4:3 TP	Transaction position. This field is used with T-count to determine whether to send all, first, middle, or last with each outbound transaction payload. System software must initialize this field with the appropriate starting value. The host controller must correctly manage this state during the lifetime of the transfer. The bit encodings are: 00 All. The entire full-speed transaction data payload is in this transaction (that is, less than or equal to 188 bytes). 01 Begin. This is the first data payload for a full-speed transaction that is greater than 188 bytes. 10 Mid. This is the middle payload for a full-speed OUT transaction that is larger than 188 bytes. 11 End. This is the last payload for a full-speed OUT transaction that was larger than 188 bytes.
2:0 T-Count	Transaction count. Software initializes this field with the number of OUT start-splits this transfer requires. Any value larger than 6 is undefined.

32.5.4.5 siTD Back Link Pointer

Doubleword 6 of an siTD is simply another schedule link pointer. This pointer is always 0 or references an siTD. This pointer cannot reference any other schedule data structure.

Table 32-56. siTD Back Link Pointer

Bit	Description
31:5 Back Pointer	This field is a physical memory pointer to an siTD.
4:1	Reserved. This field is reserved for future use. It should be cleared.
0 T	Terminate 0 siTD Back Pointer field is valid 1 siTD Back Pointer field is not valid

32.5.5 Queue Element Transfer Descriptor (qTD)

This data structure is used only with a queue head. This data structure is for one or more USB transactions. This data structure transfers as many as 20480 (5 × 4096) bytes. The structure contains two structure pointers used for queue advancement, a doubleword of transfer state, and a five-element array of data buffer pointers. This structure is 32 bytes (or one 32-byte cache line). This data structure must be physically contiguous.

The buffer associated with this transfer must be virtually contiguous. The buffer may start on any byte boundary. A separate buffer pointer list element must be used for each physical page in the buffer, regardless of whether the buffer is physically contiguous.

Host controller updates (host controller writes) to stand-alone qTDs only occur during transfer retirement. References in the following bit field definitions of updates to the qTD are to the qTD portion of a queue head.

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	offset
Next qTD Pointer																											0000				T	0x00
Alternate Next qTD Pointer																											0000				T	0x04
dt[1]	Total Bytes to Transfer[1]															IOC	C_Page[1]			Cerr[1]		PID Code	Status[1]									0x08
Buffer Pointer (Page 0)																				Current Offset[1]												0x0C
Buffer Pointer (Page 1)																				0000_0000_0000												0x10
Buffer Pointer (Page 2)																				0000_0000_0000												0x14
Buffer Pointer (Page 3)																				0000_0000_0000												0x18
Buffer Pointer (Page 4)																				0000_0000_0000												0x1C

Figure 32-48. Queue Element Transfer Descriptor (qTD)

1 Host controller read/write; all others read-only.

Queue element transfer descriptors must be aligned on 32-byte boundaries.

32.5.5.1 Next qTD Pointer

The first doubleword of an element transfer descriptor is a pointer to another transfer element descriptor.

Table 32-57. qTD Next Element Transfer Pointer (doubleword 0)

Bit	Description
31:5 Next qTD Pointer	This field contains the physical memory address of the next qTD to be processed. The field corresponds to memory address signals[31:5], respectively.
4:1	Reserved. These bits are reserved and their value has no effect on operation.
0 T	Terminate. This bit indicates to the Host Controller that there are no more valid entries in the queue. 0 Pointer is valid (points to a valid Transfer Element Descriptor). 1 Pointer is invalid.

32.5.5.2 Alternate Next qTD Pointer

The second doubleword of a queue element transfer descriptor supports hardware-only advance of the data stream to the next client buffer on short packet. To be more explicit, the host controller always uses this pointer when the current qTD is retired due to short packet.

Table 32-58. qTD Alternate Next Element Transfer Pointer (Doubleword 1)

Bit	Description
31:5 Alternate Next qTD Pointer	This field contains the physical memory address of the next qTD to be processed in the event that the current qTD execution encounters a short packet (for an IN transaction). The field corresponds to memory address signals [31:5], respectively.
4:1	Reserved. These bits are reserved and their value has no effect on operation.
0 T	Terminate. This bit indicates to the Host Controller that there are no more valid entries in the queue. 0 Pointer is valid (points to a valid Transfer Element Descriptor). 1 Pointer is invalid.

32.5.5.3 qTD Token

The third doubleword of a queue element transfer descriptor contains most of the information the host controller requires to execute a USB transaction (the remaining endpoint-addressing information is specified in the queue head). Some of the field descriptions in Table 32-59 reference fields defined in the queue head. See Section 32.5.6, “Queue Head,” for more information on these fields.

Table 32-59. qTD Token (doubleword 2)

Bit	Description
31 Data Toggle	This is the data toggle sequence bit. The use of this bit depends on the setting of the data toggle control bit in the queue head.
30:16 Total Bytes to Transfer	Total Bytes to Transfer. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction on the successful completion of the transaction. The maximum value software may store in this field is 5 × 4K (0x5000). This is the maximum number of bytes five page pointers can access. If the value of this field is 0 when the host controller fetches this transfer descriptor (and the active bit is set), the host controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for OUT transfers that total bytes to transfer be an even multiple of QH [Maximum Packet Length]. If software builds such a transfer descriptor for an OUT transfer, the last transaction is always less than QH [Maximum Packet Length]. Although it is possible to create a transfer as large as 20 KB, this assumes the page is zero. When the offset cannot be predetermined, crossing past the fifth page can be guaranteed by limiting the total bytes to 16 KB. Therefore, the maximum recommended transfer is 16 KB (0x4000).
15 IOC	Interrupt On Complete. If this bit is set, it specifies that when this qTD is completed, the host controller should issue an interrupt at the next interrupt threshold.
14:12 C_Page	Current Page. This field is used as an index into the qTD buffer pointer list. Valid values are in the range 0x0 to 0x4. The host controller is not required to write this field back when the qTD is retired.

Table 32-59. qTD Token (doubleword 2) (continued)

<table>
<tr><th>Bit</th><th colspan="2">Description</th></tr>
<tr><td>11:10
Cerr</td><td colspan="2">Error Counter. This field is a 2-bit down counter that keeps track of the number of consecutive errors detected while executing this qTD. If this field is programmed with a non-zero value during set-up, the host controller decrements the count and writes it back to the qTD if the transaction fails. If the counter counts from one to zero, the host controller marks the qTD inactive, sets the halted bit to a one, and error status bit for the error that caused Cerr to decrement to zero. An interrupt is generated if the USB error interrupt enable bit in the USB_USBINTR register is set. If the host controller driver (HCD) software programs this field to zero during set-up, the host controller does not count errors for this qTD and there is no limit on the retries of this qTD. Write-backs of intermediate execution state are to the queue head overlay area, not the qTD.</td></tr>
<tr><td></td><td>Error</td><td>Decrement Counter</td></tr>
<tr><td></td><td>Transaction Error</td><td>Yes</td></tr>
<tr><td></td><td>Data Buffer Error</td><td>No. Data buffer errors are host problems. They don't count against the device's retries. Software must not program Cerr to a value of zero when the EPS field is programmed with a value indicating a full- or low-speed device. This combination could result in undefined behavior.</td></tr>
<tr><td></td><td>Stalled</td><td>No. Detection of babble or stall automatically halts the queue head. Count is not decremented</td></tr>
<tr><td></td><td>Babble Detected</td><td>No. Detection of babble or stall automatically halts the queue head. Count is not decremented</td></tr>
<tr><td></td><td>No Error</td><td>No. If the EPS field indicates a HS device or the queue head is in the asynchronous schedule (and PIDCode indicates an IN or OUT), a bus transaction completes, and the host controller does not detect a transaction error, the host controller should reset Cerr to extend the total number of errors for this transaction. For example, Cerr should be reset with maximum value (0b11) on each successful completion of a transaction. The host controller must never reset this field if the value at the start of the transaction is 0b00.</td></tr>
<tr><td>9:8
PID Code</td><td colspan="2">This field is an encoding of the token, which should be used for transactions associated with this transfer descriptor. Encodings are:
00 OUT Token generates token (E1H)
01 IN Token generates token (69H)
10 SETUP Token generates token (2DH) (undefined if endpoint is an Interrupt transfer type, for example. µFrame S-mask field in the queue head is non-zero.)
11 Reserved</td></tr>
</table>

Table 32-59. qTD Token (doubleword 2) (continued)

<table>
<tr><th>Bit</th><th colspan="2">Description</th></tr>
<tr><td>7:0
Status</td><td colspan="2">The host controller uses this field to communicate individual command execution states back to the host controller driver (HCD) software. This field contains the status of the last transaction performed on this qTD. The bit encodings are:</td></tr>
<tr><td></td><td>Bit</td><td>Status Field Description</td></tr>
<tr><td></td><td>7</td><td>Active. Set by software to enable the execution of transactions by the host controller.</td></tr>
<tr><td></td><td>6</td><td>Halted. Set by the host controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this qTD. This can be caused by babble, the error counter counting down to zero, or reception of the STALL handshake from the device during a transaction. Any time a transaction results in the halted bit being set, the active bit is also cleared.</td></tr>
<tr><td></td><td>5</td><td>Data Buffer Error. Set by the host controller during status update to indicate the host controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, the host controller forces a time-out condition on the USB, invalidating the transaction at the source. If the host controller sets this bit to a one, it remains a one for the duration of the transfer.</td></tr>
<tr><td></td><td>4</td><td>Babble Detected. Set by the host controller during status update when babble is detected during the transaction. In addition to setting this bit, the host controller also sets the halted bit to a one. Because babble is considered a fatal error for the transfer, setting the halted bit to a one ensures no more transactions occur because of this descriptor.</td></tr>
<tr><td></td><td>3</td><td>Transaction Error (XactErr). Set by the host controller during status update in the case where the host did not receive a valid response from the device (time-out, CRC, bad PID). If the host controller sets this bit to a one, it remains a one for the duration of the transfer.</td></tr>
<tr><td></td><td>2</td><td>Missed Micro-Frame. This bit is ignored unless the QH[EPS] field indicates a full- or low-speed endpoint and the queue head is in the periodic list. This bit is set when the host controller detects a host-induced hold-off caused the host controller to miss a required complete-split transaction. If the host controller sets this bit to a one, it remains a one for the duration of the transfer.</td></tr>
<tr><td></td><td>1</td><td>Split Transaction State (SplitXstate). This bit is ignored by the host controller unless the QH[EPS] field indicates a full- or low-speed endpoint. When configured as a full- or low-speed device, the host controller uses this bit to track the state of the split- transaction. The functional requirements of the host controller for managing this state bit and the split transaction protocol depends on whether the endpoint is in the periodic or asynchronous schedule. The bit encodings are:
0 Do Start Split. This value directs the host controller to issue a start split transaction to the endpoint.
1 Do Complete Split. This value directs the host controller to issue a complete split transaction to the endpoint.</td></tr>
<tr><td></td><td>0</td><td>Ping State (P)/ERR. If the QH[EPS] field indicates a high-speed device and the PID Code indicates an OUT endpoint, this is the state bit for the ping protocol. The bit encodings are:
0 Do OUT. This value directs the host controller to issue an OUT PID to the endpoint.
1 Do Ping. This value directs the host controller to issue a PING PID to the endpoint.
If the QH[EPS] field does not indicate a high-speed device, this field is used as an error indicator bit. It is set by the host controller when a periodic split-transaction receives an ERR handshake.</td></tr>
</table>

32.5.5.4 qTD Buffer Page Pointer List

The last five doublewords of a queue element transfer descriptor is an array of physical memory address pointers. These pointers reference the individual pages of a data buffer.

System software initializes Current Offset field to the starting offset into the current page, where current page is selected via the value in the C_Page field.

Table 32-60. qTD Buffer Pointer

Bit	Name	Description
31:12	Buffer Pointer (page *n*)	Each element in the list is a 4K page aligned physical memory address. The lower 12 bits in each pointer are reserved (except for the first one), as each memory pointer must reference the start of a 4K page. The field C_Page specifies the current active pointer. When the transfer element descriptor is fetched, the starting buffer address is selected using C_Page (similar to an array index to select an array element). If a transaction spans a 4K buffer boundary, the host controller must detect the page-span boundary in the data stream, increment C_Page and advance to the next buffer pointer in the list, and conclude the transaction via the new buffer pointer.
11:0	Current Offset (Page 0)/— (Pages 1-4)	This field is reserved in all pointers except the first one (that is, Page 0). The host controller should ignore all reserved bits. For the page 0 current offset interpretation, this field is the byte offset into the current page (as selected by C_Page). The host controller is not required to write this field back when the qTD is retired. Software should ensure the reserved fields are initialized to zeros.

32.5.6 Queue Head

Figure 32-49 shows the queue head structure.

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	offset
Queue Head Horizontal Link Pointer																											00		Typ		T	0x00
RL			C	Maximum Packet Length												H	dtc	EPS		EndPt				I	Device Address							0x04[1]
Mult		Port Number						Hub Addr								µFrame C-mask								µFrame S-mask								0x08[1]
Current qTD Pointer[2]																											00000					0x0C
Next qTD Pointer[2]																											0000				T[2]	0x10[3]
Alternate Next qTD Pointer[2]																											NakCnt[2]				T[2]	0x14[3,4]
dt[1]																Total Bytes to Transfer[2]								IOC[2]		C_Page[2]						0x18[3,4]
Buffer Pointer (Page 0)[2]																								Current Offset[2]								0x1C[3,4]
Buffer Pointer (Page 1)[2]																								0000			C-prog-mask[2]					0x20[3,4]
Buffer Pointer (Page 2)[2]																								S-bytes[2]							FrameTag[2]	0x24[3,4]
Buffer Pointer (Page 3)[2]																								0000_0000_0000								0x28[3]
Buffer Pointer (Page 4)[2]																								0000_0000_0000								0x2C[3]

1 Offsets 0x04 through 0x0B contain the static endpoint state.

2 Host controller read/write; all others read-only.

3 Offsets 0x10 through 0x2F contain the transfer overlay.

4 Offsets 0x14 through 0x27 contain the transfer results.

Figure 32-49. Queue Head Layout

32.5.6.1 Queue Head Horizontal Link Pointer

The first doubleword of a queue head contains a link pointer to the next data object to be processed after any required processing in this queue has been completed, as well as the control bits defined below.

This pointer may reference a queue head or one of the isochronous transfer descriptors. It must not reference a queue element transfer descriptor.

Table 32-61. Queue Head doubleword 0

Field	Description
31:5 QHLP	Queue Head Horizontal Link Pointer. This field contains the address of the next data object to be processed in the horizontal list and corresponds to memory address signals [31:5], respectively.
4:3	Reserved. These bits must be written as zeros.

Table 32-61. Queue Head doubleword 0 (continued)

Field	Description
2:1 Typ	This field indicates to the hardware whether the item referenced by the link pointer is an iTD, siTD or a QH. This allows the host controller to perform the proper type of processing on the item after it is fetched. 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor) 11 FSTN (frame span traversal node)
0 T	Terminate. 1 Last QH (pointer is invalid). 0 Pointer is valid. If the queue head is in the context of the periodic list, a one bit in this field indicates to the host controller that this is the end of the periodic list. The host controller ignores this bit when the queue head is in the asynchronous schedule. Software must ensure that queue heads reachable by the host controller always have valid horizontal link pointers.

32.5.6.2 Endpoint Capabilities/Characteristics

The second and third doublewords of a queue head specify static information about the endpoint. This information does not change over the lifetime of the endpoint. There are three types of information in this region:

- Endpoint Characteristics. These are the USB endpoint characteristics including addressing, maximum packet size, and endpoint speed.
- Endpoint Capabilities. These are adjustable parameters of the endpoint. They affect how the endpoint data stream is managed by the host controller.
- Split Transaction Characteristics. This data structure manages full- and low-speed data streams for bulk, control, and interrupt via split transactions to USB2.0 hub transaction translator. There are additional fields for addressing the hub and scheduling the protocol transactions (for periodic).

The host controller must not modify the bits in this region.

Table 32-62. Endpoint Characteristics: Queue Head doubleword 1

Field	Description
31:28 RL	Nak Count Reload. This field contains a value used by the host controller to reload the Nak counter field.
27 C	Control Endpoint Flag. If the QH[EPS] field indicates the endpoint is not a high-speed device and the endpoint is a control endpoint, software must set this bit to a one. Otherwise, it should always set this bit to a zero.
26:16 Maximum Packet Length	This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).
15 H	Head of reclamation list flag. This bit is set by system software to mark a queue head as being the head of the reclamation list.

Table 32-62. Endpoint Characteristics: Queue Head doubleword 1 (continued)

Field	Description
14 DTC	Data Toggle Control (DTC). This bit specifies where the host controller should get the initial data toggle on an overlay transition. 0 Ignore DT bit from incoming qTD. Host controller preserves DT bit in the queue head. 1 Initial data toggle comes from incoming qTD DT bit. Host controller replaces DT bit in the queue head from the DT bit in the qTD.
13:12 EPS	Endpoint Speed. This is the speed of the associated endpoint. 00 Full-Speed (12Mbs) 01 Low-Speed (1.5Mbs) 10 High-Speed (480 Mb/s) 11 Reserved This field must not be modified by the host controller.
11:8 EndP I	Endpoint Number. This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Inactivate on next transaction. This bit is used by system software to request the host controller set the active bit to zero. This field is only valid when the queue head is in the periodic schedule and the EPS field indicates a full or low-speed endpoint. Setting this bit to a one when the queue head is in the asynchronous schedule or the EPS field indicates a high-speed device yields undefined results.
6:0 Device Address	This field selects the specific device serving as the data source or sink.

Table 32-63. Endpoint Capabilities: Queue Head doubleword 2

Field	Description
31:30 Mult	High-Bandwidth Pipe Multiplier. This field is a multiplier used to key the host controller to the number of successive packets the host controller may submit to the endpoint in the current execution. The host controller makes the simplifying assumption that software properly initializes this field (regardless of location of queue head in the schedules or other run time parameters). 00 Reserved. A zero in this field yields undefined results. 01 One transaction to be issued for this endpoint per micro-frame 10 Two transactions to be issued for this endpoint per micro-frame 11 Three transactions to be issued for this endpoint per micro-frame
29:23 Port Number	This field is ignored by the host controller unless the EPS field indicates a full- or low-speed device. The value is the port number identifier on the USB 2.0 hub (for hub at device address Hub Addr below), below which the full- or low-speed device associated with this endpoint is attached. This information is used in the split-transaction protocol.
22:16 Hub Addr	This field is ignored by the host controller unless the EPS field indicates a full-or low-speed device. The value is the USB device address of the USB 2.0 hub below which the full- or low-speed device associated with this endpoint is attached. This field is used in the split-transaction protocol.

Table 32-63. Endpoint Capabilities: Queue Head doubleword 2 (continued)

Field	Description
15:8 µFrame C-mask	This field is ignored by the host controller unless the EPS field indicates this device is a low- or full-speed device and this queue head is in the periodic list. This field (along with the active and SplitX-state fields) is used to determine during which micro-frames the host controller should execute a complete-split transaction. When the criteria for using this field are met, a zero value in this field has undefined behavior. This field is used by the host controller to match against the three low-order bits of the USB_FRINDEX register. If the USB_FRINDEX register bits decode to a position where the µFrame C- mask field is a one, this queue head is a candidate for transaction execution. There may be more than one bit in this mask set.
7:0 µFrame S-mask	Interrupt Schedule Mask. This field is used for all endpoint speeds. Software should set this field to a zero when the queue head is on the asynchronous schedule. A non-zero value in this field indicates an interrupt endpoint. The host controller uses the value of the three low-order bits of the USB_FRINDEX register as an index into a bit position in this bit vector. If the µFrame S-mask field has a one at the indexed bit position, this queue head is a candidate for transaction execution. If the EPS field indicates the endpoint is a high-speed endpoint, the transaction executed is determined by the PID_Code field contained in the execution area. This field is also used to support split transaction types such as Interrupt (IN/OUT). This condition is true when this field is non-zero and the EPS field indicates this is either a full- or low-speed device. A zero value in this field, in combination with existing in the periodic frame list, has undefined results.

32.5.6.3 Transfer Overlay

The nine doublewords in this area represent a transaction working space for the host controller. The general operational model is that the host controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, it follows the queue head horizontal link pointer to the next queue head. The host controller never follows the next transfer queue element or alternate queue element pointers unless it is actively attempting to advance the queue. For the duration of the transfer, the host controller keeps the incremental status of the transfer in the overlay area. When the transfer is complete, the results are written back to the original queue element.

The doubleword of a queue head contains a pointer to the source qTD currently associated with the overlay. The host controller uses this pointer to write back the overlay area into the source qTD after the transfer is complete.

Table 32-64. Current qTD Link Pointer

Field	Description
31:5 Current qTD Pointer	Current Element Transaction Descriptor Link Pointer. This field contains the address Of the current transaction being processed in this queue and corresponds to memory address signals [31:5], respectively.
4:0	Reserved. These bits are ignored by the host controller when using the value as an address to write data. The actual value may vary depending on the usage.

The doublewords 4-11 of a queue head are the transaction overlay area. This area has the same base structure as a queue element transfer descriptor. The queue head utilizes the reserved fields of the page pointers to implement tracking the state of split transactions.

This area is characterized as an overlay because when the queue is advanced to the next queue element, the source queue element is merged onto this area. This area serves an execution cache for the transfer.

Table 32-65. Host-Controller Rules for Bits in Overlay (doublewords 5, 6, 8 and 9)

doubleword	QH Offset	Bit	Description
5	0x14	4:1 NakCnt	Nak counter—RW. This field is a counter the host controller decrements when a transaction for the endpoint associated with this queue head results in a Nak or Nyet response. This counter is reloaded from RL before a transaction is executed during the first pass of the reclamation list (relative to an asynchronous list restart condition). It is also loaded from RL during an overlay.
6	0x18	31 DT	The data toggle control controls whether the host controller preserves this bit when an overlay operation is performed.
6	0x18	15 IOC	Interrupt on complete. The IOC control bit is always inherited from the source qTD when the overlay operation is performed.
6	0x18	11:10 Cerr	Error counter. This two-bit field is copied from the qTD during the overlay and written back during queue advancement.
6	0x18	0 Status	Ping state (P)/ERR. If the EPS field indicates a high-speed endpoint, then this field should be preserved during the overlay operation.
8	0x20	7:0 C-prog-mask	Split-transaction complete-split progress. This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	0x24	11:5 S-bytes	Software must ensure that the S-bytes field in a qTD is zero before activating the qTD. This field is used to keep track of the number of bytes sent or received during an IN or OUT split transaction.
9	0x24	4:0 FrameTag	Split-transaction frame tag. This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.

32.5.7 Periodic Frame Span Traversal Node (FSTN)

This data structure is only for managing full- and low-speed transactions that span a host-frame boundary. Software must not use an FSTN in the asynchronous schedule. An FSTN in the asynchronous schedule results in undefined behavior. Software must not use the FSTN feature with a host controller whose HCIVERSION register indicates a revision implementation below 0x0096. FSTNs were not defined for EHCI implementations before revision 0.96 of the EHCI specification and their use may yield undefined results.

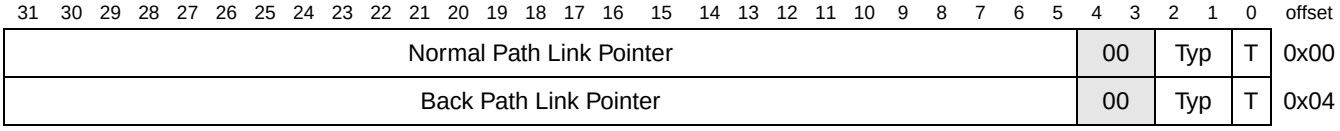


Figure 32-50. Frame Span Traversal Node Structure

32.5.7.1 FTSN Normal Path Pointer

The first doubleword of an FSTN contains a link pointer to the next schedule object. This object can be of any valid periodic schedule data type.

Table 32-66. FTSN Normal Path Pointer

Bit	Description
31:5 NPLP	Normal Path Link Pointer. This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4:3	Reserved. These bits must be written as 0s.
2:1 Typ	This field indicates to the host controller whether the item referenced is an iTD/siTD, a QH, or an FSTN. This allows the host controller to perform the proper type of processing on the item after it is fetched. 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor) 11 FSTN (Frame Span Traversal Node)
0 T	Terminate. 0 Link Pointer is valid. 1 Link Pointer field is not valid.

32.5.7.2 FSTN Back Path Link Pointer

The second doubleword of an FTSN node contains a link pointer to a queue head. If the T-bit in this pointer is a zero, then this FSTN is a save-place indicator. Its Typ field must be set by software to indicate the target data structure is a queue head. If the T-bit in this pointer is set, this FSTN is the restore indicator. When the T-bit is a one, the host controller ignores the Typ field.

Table 32-67. FSTN Back Path Link Pointer

Bit	Description
31:5 BPLP	Back Path Link Pointer. This field contains the address of a queue head. This field corresponds to memory address signals [31:5], respectively.
4:3	Reserved. These bits must be written as 0s.
2:1 Typ	Software must ensure this field is set to indicate the target data structure is a queue head (01). Any other value in this field yields undefined results.
0 T	Terminate. 0 Link Pointer is valid (that is, the host controller may use bits [31:5] [in combination with the CTRLDSSEGMENT register if applicable] as a valid memory address). This value also indicates that this FSTN is a save-place indicator. 1 Link Pointer field is not valid (that is, the host controller must not use bits [31:5] [in combination with the CTRLDSSEGMENT register if applicable] as a valid memory address). This value also indicates that this FSTN is a restore indicator.

32.6 Host Operational Model

The general operational model is for the USB modules in host mode is defined by the enhanced host controller interface (EHCI) specification. The EHCI specification describes the register-level interface for a host controller for the USB Revision 2.0. It includes a description of the hardware/software interface between system software and host controller hardware. Information concerning the initialization of the USB modules is included in the following section; however, the full details of the EHCI specification are beyond the scope of this document.

32.6.1 Host Controller Initialization

After initial power-on or HCReset (hardware or via HCReset bit in the USB_USBCMD register), all of the operational registers are at their default values, as illustrated in Table 32-68. After a hardware reset, only the operational registers not contained in the auxiliary power well is at their default values.

Table 32-68. Default Values of Operational Register Space

Operational Register	Default Value (after Reset)
USB_USBCMD	0x0008_0000 (0x0008_0B00 if Asynchronous Schedule Park Capability is set)
USB_USBSTS	0x0000_1000
USB_USBINTR	0x0000_0000
USB_FRINDEX	0x0000_0000
CTRLDSSEGMENT	0x0000_0000
USB_PERIODICLISTBASE	Undefined
USB_ASYNCLISTADDR	Undefined
USB_CONFIGFLAG	0x0000_0000
USB_PORTSC*n*	0x0000_2000 (w/PPC set); 0x0000_3000 (w/PPC cleared)

To initialize the host controller, software should perform the following steps:

1. Optionally set streaming disable in the USB_USBMODE register.
2. Optionally modify the USB_BURSTSIZE register.
3. Program the PTS field of the USB_PORTSC*n* register if using a non-ULPI PHY.
4. Set the USB_EN bit in the CONTROL register.
5. Program the CTRLDSSEGMENT register with 4-Gigabyte segment where all of the interface data structures are allocated.
6. Write the appropriate value to the USB_USBINTR register to enable the appropriate interrupts.
7. Write the base address of the periodic frame list to the PERIODICLIST BASE register. If there are no work items in the periodic schedule, all elements of the periodic frame list should have their T-Bits set.
8. Write the USB_USBCMD register to set the desired interrupt threshold, frame list size (if applicable), and turn the controller by setting the run/stop bit.

At this point, the host controller is running and the port registers begin reporting device connects. System software can enumerate a port through the reset process (where the port is in the enabled state). At this point, the port is active with SOFs occurring down the enabled port enabled high-speed ports, but the schedules have not yet been enabled. The EHCI host controller does not transmit SOFs to enabled full- or low-speed ports.

To communicate with devices via the asynchronous schedule, system software must write the ASYNDLISTADDR register with the address of a control or bulk queue head. Software must then enable the asynchronous schedule by writing a one to the asynchronous schedule enable bit in the

USB_USBCMD register. To communicate with devices via the periodic schedule, system software must enable the periodic schedule by writing a one to the periodic schedule enable bit in the USB_USBCMD register. The schedules can be turned on before the first port is reset (and enabled).

Any time the USB_USBCMD register is written, system software must ensure the appropriate bits are preserved, depending on the intended operation.

32.6.1.1 Port Power

The port power control (PPC) bit in the USB_HCSPARAMS register indicates whether the USB 2.0 host controller has port power control. When the PPC bit is a one, the host controller supports port power switches. Each available switch has an output enable. PPE is controlled based on the state of the combination bits PPC bit, EHCI configured (CF) bit, and individual port power (PP) bits.

32.6.1.2 Reporting Over-Current

Host ports by definition are power providers on USB. Whether the ports are considered high- or low-powered is a platform implementation issue. Each EHCI USB_PORTSC*n* register has an over-current status and over-current change bit. The functionality of these bits is specified in the USB Specification Revision 2.0.

32.6.2 Suspend/Resume

The host controller provides an equivalent suspend and resume model as that defined for individual ports in a USB 2.0 hub. Control mechanisms are provided to allow system software to suspend and resume individual ports. The mechanisms allow the individual ports to be resumed completely via software initiation. Other control mechanisms are provided to parameterize the host controller's response (or sensitivity) to external resume events. In this discussion, host-initiated or software-initiated resumes are called resume events/actions; bus-initiated resume events are called wake-up events. The classes of wakeup events are:

- Remote-wakeup enabled device asserts resume signaling. Similar to USB 2.0 hubs, when in host mode, the host controller responds to explicit device, resumes signaling, and wakes up the system (if necessary).
- Port connect and disconnect. Sensitivity to these events can be turned on or off by using the port control bits in the USB_PORTSC*n* register. An over-current event does not wake the USB core.

Selective suspend is a feature supported by the USB_PORTSC*n* register. It places specific ports into a suspend mode. This feature is a functional component for implementing the appropriate power management policy implemented in a particular operating system. When system software intends to suspend the bus, it should suspend the enabled port, then shut off the controller by setting the run/stop bit in the USB_USBCMD register to a zero.

When a wake event occurs, the system resumes operation and system software must set the run/stop bit to a one and resume the suspended port.

32.6.2.1 Port Suspend/Resume

System software places the USB into suspend mode by writing a one into the appropriate USB_PORTSC*n* suspend bit. Software must only set the suspend bit when the port is in the enabled state (port enabled bit is a one).

The host controller may evaluate the suspend bit immediately or wait until a micro-frame or frame boundary occurs. If evaluated immediately, the port is not suspended until the current transaction (if one is executing) completes. Therefore, there may be several micro-frames of activity on the port until the host controller evaluates the suspend bit. The host controller must evaluate the suspend bit at least every frame boundary.

System software can initiate a resume on the suspended port by writing a one to the force port resume bit. Software should not attempt to resume a port unless the port reports that it is in the suspended state. If system software sets the force port resume bit when the port is not in the suspended state, the resulting behavior is undefined. To assure proper USB device operation, software must wait for at least 10 milliseconds after a port indicates it is suspended (suspend bit is a one) before initiating a port resume via the force port resume bit. When force port resume bit is set, the host controller sends resume signaling down the port. System software times the duration of the resume (nominally 20 milliseconds), and then clears the force port resume bit. When the host controller receives the write to transition force port resume to zero, it completes the resume sequence as defined in the USB specification, and clears both the force port resume and suspend bits. Software-initiated port resumes do not affect the port change detect bit in the USB_USBSTS register nor do they cause an interrupt if the port change interrupt enable bit in the USB_USBINTR register is a one. When a wake event occurs on a suspended port, the resume signaling is detected by the port and the resume is reflected downstream within 100 µsec. The port's force port resume bit is set and the port change detect bit in the USB_USBSTS register is set. If the port change interrupt enable bit in the USB_USBINTR register is a one, the host controller issues a hardware interrupt.

System software observes the resume event on the port, delays a port resume time (nominally 20 milliseconds), and then terminates the resume sequence by clearing the force port resume bit in the port. The host controller receives the write of zero to force port resume, terminates the resume sequence, and clears the force port resume and suspend port bits. Software can determine the port is enabled (not suspended) by sampling the USB_PORTSC*n* register and observing that the suspend and force port resume bits are zero. Software must ensure that the host controller is running (HCHalted bit in the USB_USBSTS register is a zero), before terminating a resume by clearing the port's force port Resume bit. If HCHalted is a one when force port resume is cleared, SOFs does not occur down the enabled port and the device returns to suspend mode in a maximum of 10 milliseconds.

Table 32-69 summarizes the wake-up events. When a resume event is detected, the port change detect bit in the USB_USBSTS register is set. If the port change interrupt enable bit is a one in the USB_USBINTR register, the host controller also generates an interrupt on the resume event. Software acknowledges the resume event interrupt by clearing the port change detect status bit in the USB_USBSTS register.

Table 32-69. Behavior During Wake-up Events

Port Status and Signaling Type	Signaled Port Response	Device State	
		D0	not D0
Port disabled, resume K-State received	No effect.	N/A	N/A
Port suspended, Resume K-State received	Resume reflected downstream on signaled port. Force port resume status bit in USB_PORTSC*n* register is set. Port change detect bit in USB_USBSTS register is set.	[1], [2]	2
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is set. A disconnect is detected.	Depending in the initial port state, the USB_PORTSC*n* connect and enable status bits are cleared, and the connect change status bit is set. Port change detect bit in the USB_USBSTS register is set.	[1], 2	2
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is cleared. A disconnect is detected.	Depending on the initial port state, the USB_PORTSC*n* connect and enable status bits are cleared, and the connect change status bit is set. Port change detect bit in the USB_USBSTS register is set.	[1], 3	3
Port is not connected and the port's WKCNNT_E bit is a one. A connect is detected.	USB_PORTSC*n* connect status and connect status change bits are set. Port change detect bit in the USB_USBSTS register is set.	[1], 2	2
Port is not connected and the port's WKCNNT_E bit is a zero. A connect is detected.	USB_PORTSC*n* connect status and connect status change bits are set. Port change detect bit in the USB_USBSTS register is set.	[1], 3	3
Port is connected and the port's WKOC_E bit is a one. An over-current condition occurs.	USB_PORTSC*n* over-current active, over-current change bits are set. If port enable/disable bit is a one, it is cleared. Port change detect bit in the USB_USBSTS register is set	[1], 2	2
Port is connected and the port's WKOC_E bit is a zero. An over-current condition occurs.	USB_PORTSC*n* over-current active, over-current change bits are set. If port enable/disable bit is a one, it is cleared. Port change detect bit in the USB_USBSTS register is set.	[1], 3	3

1 Hardware interrupt issued if port change interrupt enable bit in the USB_USBINTR register is set.

2 ME# asserted if enabled. PPME Status must always be set.

3 PME# not asserted.

32.6.3 Schedule Traversal Rules

The host controller executes transactions for devices using a simple, shared-memory schedule. The schedule is comprised of a few data structures, organized into two distinct lists. The data structures provide the maximum flexibility required by USB, minimize memory traffic, and hardware/software complexity.

System software maintains two schedules for the host controller: a periodic schedule and an asynchronous schedule. The root of the periodic schedule is the USB_PERIODICLISTBASE register. See Section 32.2.4.6, “Periodic Frame List Base Address (USB_PERIODICLISTBASE) Register,” for more

information. The USB_PERIODICLISTBASE register is the physical memory base address of the periodic frame list. The periodic frame list is an array of physical memory pointers. The objects referenced from the frame list must be valid schedule data structures as defined in Section 32.5, “Host Data Structures.” In each micro-frame, if the periodic schedule is enabled, then the host controller must execute from the periodic schedule before executing from the asynchronous schedule. It only executes from the asynchronous schedule after it encounters the end of the periodic schedule. The host controller traverses the periodic schedule by constructing an array offset references from the USB_PERIODICLISTBASE and the USB_FRINDEX registers (see Figure 32-51). It fetches the element and begins traversing the graph of linked schedule data structures.

The end of the periodic schedule is identified by a next link pointer of a schedule data structure having its T-bit set. When the host controller encounters a T-Bit set during a horizontal traversal of the periodic list, it interprets this as an End-Of-Periodic-List mark. This causes the host controller to cease working on the periodic schedule and transition immediately to traversing the asynchronous schedule. After this transition is made, the host controller executes from the asynchronous schedule until the end of the micro-frame.

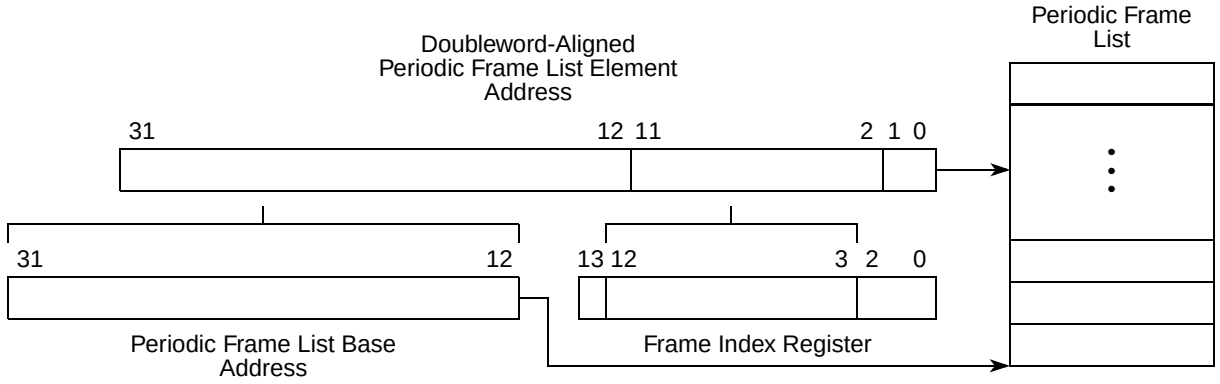


Figure 32-51. Derivation of Pointer into Frame List Array

When the host controller determines it is time to execute from the asynchronous list, it uses the operational USB_ASYNCLISTADDR register to access the asynchronous schedule, as shown in Figure 32-52.

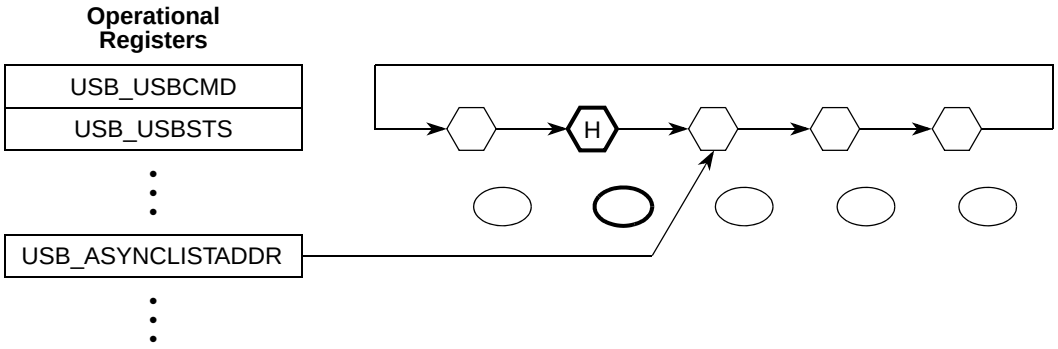


Figure 32-52. General Format of Asynchronous Schedule List

The USB_ASYNCLISTADDR register contains a physical memory pointer to the next queue head. When the host controller makes a transition to executing the asynchronous schedule, it begins by reading the

queue head referenced by the USB_ASYNCLISTADDR register. Software must set queue head horizontal pointer T-bits to a zero for queue heads in the asynchronous schedule.

32.6.4 Periodic Schedule Frame Boundaries vs. Bus Frame Boundaries

The USB specification revision 2.0 requires that the frame boundaries (SOF frame number changes) of the high-speed bus and the full- and low-speed bus(es) below USB 2.0 hubs be strictly aligned. Super-imposed on this requirement is USB 2.0 hubs manage full- and low-speed transactions via a micro-frame pipeline (see start- (SS) and complete- (CS) splits illustrated in Figure 32-53). A simple, direct projection of the frame boundary model into the host controller interface schedule architecture creates tension (complexity for both hardware and software) between the frame boundaries and the scheduling mechanisms required to service the full- and low-speed transaction translator periodic pipelines.

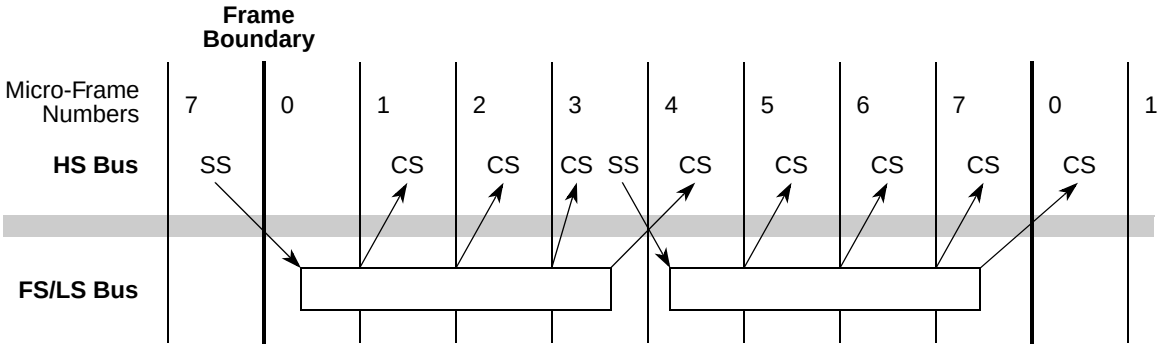


Figure 32-53. Frame Boundary Relationship Between HS Bus and FS/LS Bus

The simple projection, as Figure 32-53 illustrates, introduces frame-boundary wrap conditions for scheduling on both the beginning and end of a frame. To reduce the complexity for hardware and software, the host controller is required to implement a one micro-frame phase shift for its view of frame boundaries. The phase shift eliminates the beginning of frame and frame-wrap scheduling boundary conditions.

The implementation of this phase shift requires the host controller use one register value for accessing the periodic frame list and another value for the frame number value included in the SOF token. These two values are separate, but tightly coupled. The periodic frame list is accessed via the USB_FRINDEX register. The USB_FRINDEX[2:0] bits represent the micro-frame number. The SOF value is coupled to the value of USB_FRINDEX[13:3]. USB_FRINDEX[13:3] and the SOF value are incremented based on USB_FRINDEX[2:0]. The SOF must value be delayed from the USB_FRINDEX value by one micro-frame. The one micro-frame delay yields a host controller periodic schedule and bus frame boundary relationship as illustrated in Figure 32-54. This adjustment allows software to trivially schedule the periodic start and complete-split transactions for full-and low-speed periodic endpoints, using the natural alignment of the periodic schedule interface.

Figure 32-54 illustrates how periodic schedule data structures relate to schedule frame boundaries and bus frame boundaries. To aid the presentation, two terms are defined. The host controller's view of the 1-millisecond boundaries is called H-Frames. The high-speed bus's view of the 1-millisecond boundaries is called B-Frames.

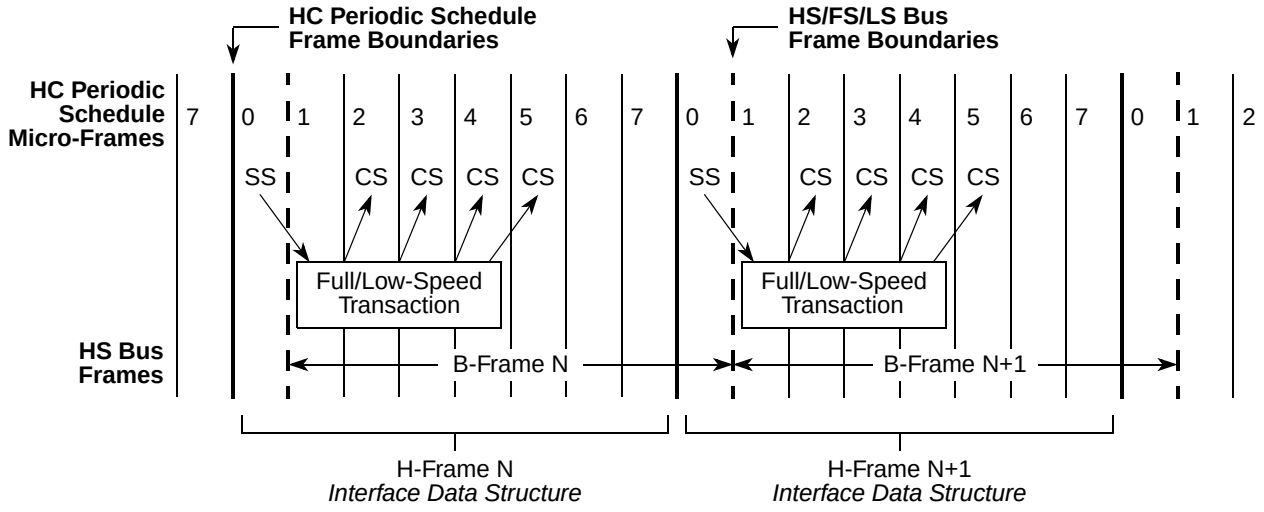


Figure 32-54. Relationship of Periodic Schedule Frame Boundaries to Bus Frame Boundaries

H-Frame boundaries for the host controller correspond to increments of USB_FRINDEX[13:3]. Micro-frame numbers for the H-Frame are tracked by USB_FRINDEX[2:0]. B-Frame boundaries are visible on the high-speed bus via changes in the SOF token's frame number. Micro-frame numbers on the high-speed bus are only derived from the SOF token's frame number (the high-speed bus sees eight SOFs with the same frame number value). H-Frames and B-Frames have the fixed relationship (B-Frames lag H-Frames by one micro-frame time) illustrated in Figure 32-54. The host controller's periodic schedule is naturally aligned to H-Frames. Software schedules transactions for full- and low-speed periodic endpoints relative the H-Frames. The result is these transactions execute on the high-speed bus at exactly the right time for the USB 2.0 hub periodic pipeline. As described in Section 32.2.4.4, “USB Frame Index (USB_FRINDEX) Register,” the SOF value can be implemented as a shadow register (in this example, called SOFV), which lags the USB_FRINDEX[13:3] bits by one micro-frame count. Table 32-70 illustrates the required relationship between the value of USB_FRINDEX and the value of SOFV. This lag behavior can be accomplished by incrementing USB_FRINDEX[13:3] based on carry-out of the 7 to 0 increment of USB_FRINDEX[2:0] and incrementing SOFV based on the transition of 0 to 1 of USB_FRINDEX[2:0].

Software can write to USB_FRINDEX. Section 32.2.4.4, “USB Frame Index (USB_FRINDEX) Register,” provides the requirements that software should adhere to when writing a new value in USB_FRINDEX.

Table 32-70. Operation of USB_FRINDEX and SOFV (SOF Value Register)

Current			Next		
USB_FRINDEX[13:3]	SOFV	USB_FRINDEX[2:0]	USB_FRINDEX[13:3]	SOFV	USB_FRINDEX[2:0]
N	N	111	N+1	N	000
N+1	N	000	N+1	N+1	001
N+1	N+1	001	N+1	N+1	010
N+1	N+1	010	N+1	N+1	011
N+1	N+1	011	N+1	N+1	100
N+1	N+1	100	N+1	N+1	101
N+1	N+1	101	N+1	N+1	110
N+1	N+1	110	N+1	N+1	111

32.6.5 Periodic Schedule

The periodic schedule traversal is enabled or disabled via the periodic schedule enable bit in the USB_USBCMD register. If the periodic schedule enable bit is cleared, the host controller simply does not try to access the periodic frame list via the USB_PERIODICLISTBASE register. Likewise, when the periodic schedule enable bit is a one, the host controller uses the USB_PERIODICLISTBASE register to traverse the periodic schedule. The host controller does not react to modifications to the periodic schedule enable immediately. To eliminate conflicts with split transactions, the host controller evaluates the periodic schedule enable bit only when USB_FRINDEX[2:0] is 0b000. System software must not disable the periodic schedule if the schedule contains an active split transaction work item that spans the 0b000 micro-frame. These work items must be removed from the schedule before the periodic schedule enable bit is cleared. The periodic schedule status bit in the USB_USBSTS register indicates status of the periodic schedule. System software enables (or disables) the periodic schedule by setting (or clearing) the periodic schedule enable bit in the USB_USBCMD register. Software then can poll the periodic schedule status bit to determine when the periodic schedule has made the desired transition. Software must not modify the periodic schedule enable bit unless the value of the periodic schedule enable bit equals that of the periodic schedule status bit.

The periodic schedule manages all isochronous and interrupt transfer streams. The base of the periodic schedule is the periodic frame list. Software links schedule data structures to the periodic frame list to produce a graph of scheduled data structures. The graph represents an appropriate sequence of transactions on the USB. Figure 32-55 illustrates isochronous transfers (using iTDs and siTDs) with a period of one are linked directly to the periodic frame list. Interrupt transfers (managed with queue heads) and isochronous streams with periods other than one are linked following the period-one iTD/siTDs. Interrupt queue heads are linked into the frame list ordered by poll rate. Longer poll rates are linked first (for example, closest to the periodic frame list), followed by shorter poll rates, with queue heads with a poll rate of one, on the end.

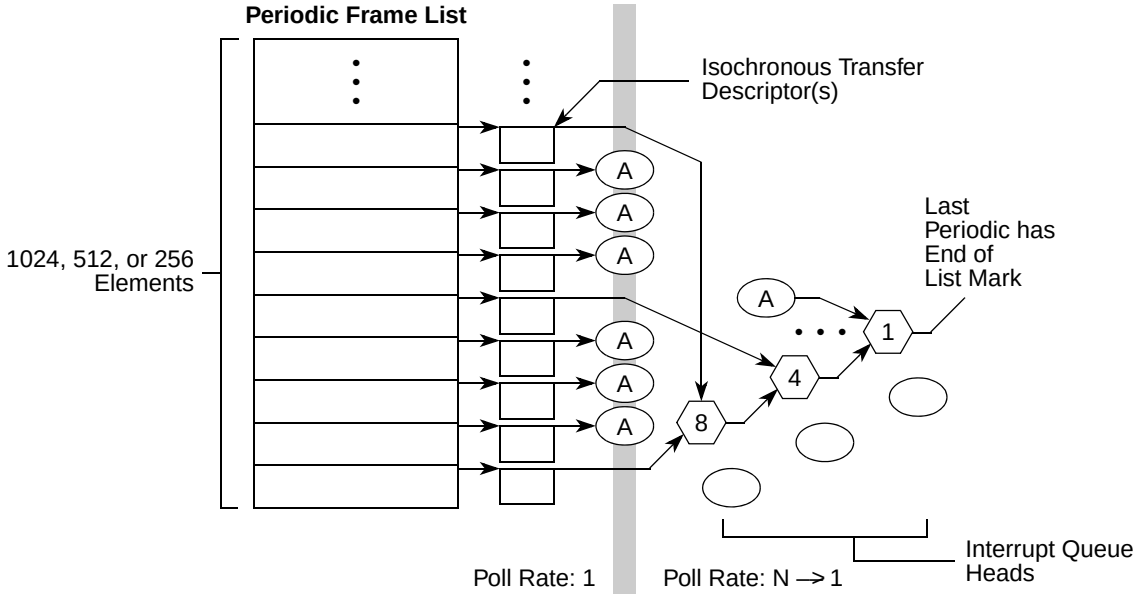


Figure 32-55. Example Periodic Schedule

32.6.6 Managing Isochronous Transfers Using iTDs

The structure of an iTD is presented in Section 32.5.3, "Isochronous (High-Speed) Transfer Descriptor (iTD)." There are four distinct sections to an iTD:

- The first field is the next link pointer. This is for schedule linkage purposes only.
- Transaction description array. This area is an eight-element array. Each element represents control and status information for one micro-frame's worth of transactions for a single high-speed isochronous endpoint.
- The buffer page pointer array is a 7-element array of physical memory pointers to data buffers. These are 4K aligned pointers to physical memory.
- Endpoint capabilities. This area utilizes the unused low-order 12 bits of the buffer page pointer array. The fields in this area are used across all transactions executed for this iTD, including endpoint addressing, transfer direction, maximum packet size, and high-bandwidth multiplier.

32.6.6.1 Host Controller Operational Model for iTDs

The host controller uses USB_FRINDEX[12:3] to index into the periodic frame list. This means the host controller visits each frame list element eight consecutive times before incrementing to the next periodic frame list element. Each iTD contains eight transaction descriptions that map directly to USB_FRINDEX[2:0]. Each iTD can span eight micro-frames worth of transactions. When the host controller fetches an iTD, it uses USB_FRINDEX[2:0] to index into the transaction description array. If the active bit in the status field of the indexed transaction description is cleared, the host controller ignores the iTD and follows the next pointer to the next schedule data structure.

When the indexed active bit is a one, the host controller continues to parse the iTD. It stores the indexed transaction description and the general endpoint information (device address, endpoint number, maximum

packet size, etc.). It also uses the page select (PG) field to index the buffer pointer array, storing the selected buffer pointer, and the next sequential buffer pointer. For example, if PG field is a 0, the host controller stores Page 0 and Page 1.

The host controller constructs a physical data buffer address by concatenating the current buffer pointer (as selected using the current transaction description's PG field) and the transaction description's transaction offset field. The host controller uses the endpoint addressing information and I/O-bit to execute a transaction to the appropriate endpoint. When the transaction is complete, the host controller clears the active bit and writes back any additional status information to the status field in the currently selected transaction description.

The data buffer associated with the iTD must be virtually contiguous memory. Seven page pointers are provided to support eight high-bandwidth transactions regardless of the starting packet’s offset alignment into the first page. A starting buffer pointer (physical memory address) is constructed by concatenating the page pointer (example: page 0 pointer) selected by the active transaction descriptions’ PG (example value: 0b00) field with the transaction offset field. As the transaction moves data, the host controller must detect when an increment of the current buffer pointer crosses a page boundary. When this occurs, the host controller replaces the current buffer pointer’s page portion with the next page pointer (example: page 1 pointer) and continues to move data. The size of each bus transaction is determined by the value in the maximum packet size field. An iTD supports high-bandwidth pipes via the mult (multiplier) field. When the mult field is 1, 2, or 3, the host controller executes the specified number of maximum packet sized bus transactions for the endpoint in the current micro-frame. In other words, the mult field represents a transaction count for the endpoint in the current micro-frame. If the mult field is zero, the operation of the host controller is undefined. The transfer description services all transactions indicated by the mult field.

For OUT transfers, the value of the transaction *n* length field represents the total bytes to be sent during the micro-frame. Software must set the mult field to be consistent with transaction *n* length and maximum packet size. The host controller sends the bytes in maximum packet sized portions. After each transaction, the host controller decrements its local copy of transaction *n* length by maximum packet size. The number of bytes the host controller sends is always maximum packet size or transaction *n* length, whichever is less. The host controller advances the transfer state in the transfer description, updates the appropriate record in the iTD, and moves to the next schedule data structure. The maximum sized transaction supported is 3 × 1024 bytes.

For IN transfers, the host controller issues mult transactions. It is assumed that software has properly initialized the iTD to accommodate all of the possible data. During each IN transaction, the host controller must use maximum packet size to detect packet babble errors. The host controller keeps the sum of bytes received in the transaction *n* length field. After all transactions for the endpoint have completed for the micro-frame, transaction *n* length contains the total bytes received. If the final value of transaction *n* length is less than the value of maximum packet size, less data than was allowed for was received from the associated endpoint. This short packet condition does not set the USBINT bit in the USB_USBSTS register. The host controller does not detect this condition. If the device sends more than transaction *n* length or maximum packet size bytes (whichever is less), the host controller sets the babble detected bit and clears the active bit. The host controller is not required to update the iTD field transaction *n* length in this error scenario. If the mult field is greater than one, the host controller automatically executes the value of mult transactions. The host controller does not execute all mult transactions if:

- The endpoint is an OUT and Transaction *n* Length goes to zero before all the Mult transactions have executed (ran out of data), or
- The endpoint is an IN and the endpoint delivers a short packet, or an error occurs on a transaction before Mult transactions have been executed. The end of micro-frame may occur before all of the transaction opportunities have been executed. When this happens, the transfer state of the transfer description is advanced to reflect the progress that was made, the result written back to the iTD and the host controller proceeds to processing the next micro-frame.

32.6.6.2 Software Operational Model for iTDs

A client buffer request to an isochronous endpoint may span 1 to N micro-frames. When N is larger than one, system software may have to use multiple iTDs to read or write data with the buffer (if N is larger than eight, it must use more than one iTD).

Figure 32-56 illustrates the simple model of how a client buffer is mapped by system software to the periodic schedule (the periodic frame list and a set of iTDs). On the right of Figure 32-56 is the client description of its request. The description includes a buffer base address plus additional annotations to identify which portions of the buffer should be used with each bus transaction. In the middle of Figure 32-56 are the iTD data structures used by the system software to service the client request. Each iTD can be initialized to service as many as 24 transactions, organized into eight groups of as many as three transactions each. Each group maps to one micro-frame's worth of transactions. The EHCI controller does not provide per-transaction results within a micro-frame. It treats the per-micro-frame transactions as a single logical transfer. On the left of Figure 32-56 is the host controller’s frame list. System software establishes references from the appropriate locations in the frame list to each of the appropriate iTDs. If the buffer is large, system software can use a small set of iTDs to service the entire buffer. System software can activate the transaction description records (contained in each iTD) in any pattern required for the particular data stream.

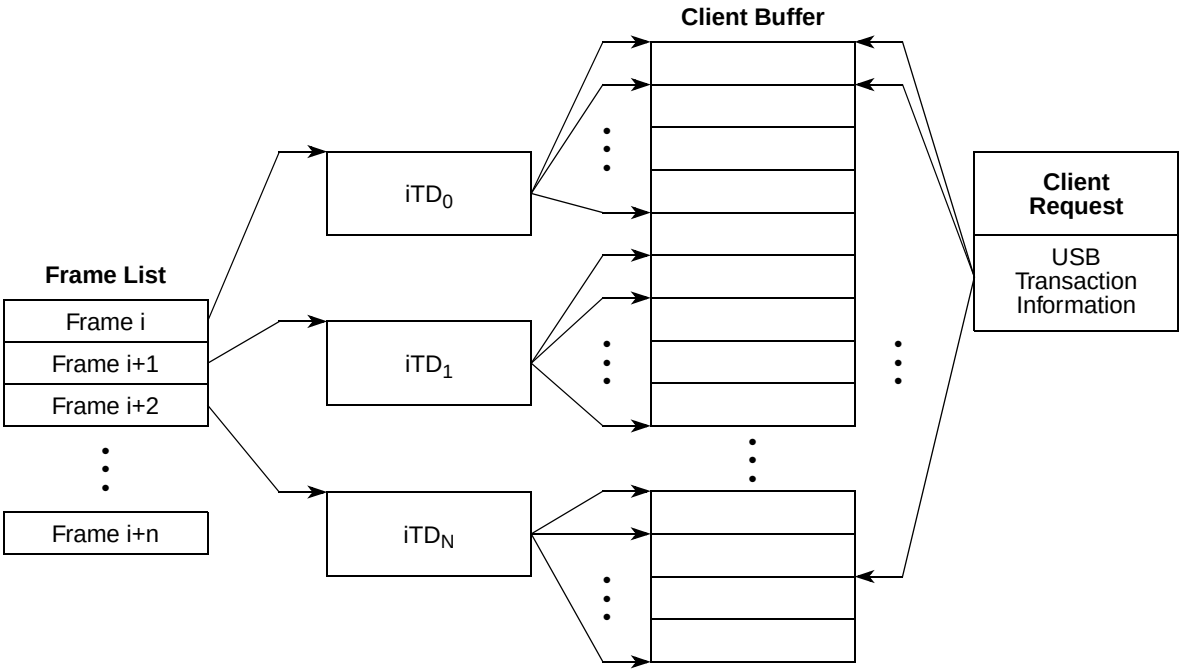


Figure 32-56. Example Association of iTDs to Client Request Buffer

As noted earlier, the client request includes a pointer to the base of the buffer and offsets into the buffer to annotate which buffer sections are used on each bus transaction that occurs on this endpoint. System software must initialize each transaction description in an iTD to ensure it uses the correct portion of the client buffer. For example, for each transaction description, the PG field is set to index the correct physical buffer page pointer and the transaction offset field is set relative to the correct buffer pointer page (for example, the same one referenced by the PG field). When the host controller executes a transaction, it selects a transaction description record based on USB_FRINDEX[2:0]. It then uses the current page buffer pointer (as selected by the PG field) and concatenates to the transaction offset field. The result is a starting buffer address for the transaction. As the host controller moves data for the transaction, it must watch for a page wrap condition and properly advance to the next available page buffer pointer. System software must not use the Page 6 buffer pointer in a transaction description where the length of the transfer wraps a page boundary. Doing so yields undefined behavior. The host controller hardware is not required to alias the page selector to page zero. USB 2.0 isochronous endpoints can specify a period greater than one. Software can achieve the appropriate scheduling by linking iTDs into the appropriate frames (relative to the frame list) and by setting appropriate transaction description elements active bits to a one.

32.6.6.2.1 Periodic Scheduling Threshold

The isochronous scheduling threshold field in the USB_HCCPARAMS register is an indicator to system software as to how the host controller pre-fetches and effectively caches schedule data structures. System software uses it when adding isochronous work items to the periodic schedule. The value of this field indicates to system software the minimum distance it can update isochronous data (relative to the current location of the host controller execution in the periodic list) and continue to have the host controller process them.

The iTD and siTD data structures each describe eight micro-frames worth of transactions. The host controller is allowed to cache one (or more) of these data structures to reduce memory traffic. There are three basic caching models that account for the fact the isochronous data structures span eight micro-frames. The three caching models are: no caching, micro-frame caching and frame caching.

When software is adding new isochronous transactions to the schedule, it always performs a read of the USB_FRINDEX register to determine the current frame and micro-frame the host controller is currently executing. Of course, there is no information about where in the micro-frame the host controller is, so a constant uncertainty factor of one micro-frame has to be assumed. Combining the knowledge of where the host controller is executing with the knowledge of the caching model allows the definition of simple algorithms for how closely software can reliably work to the executing host controller.

No caching is indicated with a value of zero in the isochronous scheduling threshold field. The host controller may pre-fetch data structures during a periodic schedule traversal (per micro-frame), but always dumps any accumulated schedule state at the end of the micro-frame. At the appropriate time relative to the beginning of every micro-frame, the host controller always begins schedule traversal from the frame list. Software can use the value of the USB_FRINDEX register (plus the constant 1 uncertainty-factor) to determine the approximate position of the executing host controller. When no caching is selected, software can add an isochronous transaction as near as two micro-frames in front of the current executing position of the host controller.

Frame caching is indicated with a non-zero value in bit [7] of the isochronous scheduling threshold field. In the frame-caching model, system software assumes the host controller caches one (or more) isochronous data structures for an entire frame (8 micro-frames). Software uses the value of the USB_FRINDEX register (plus the constant 1 uncertainty) to determine the current micro-frame/frame (assume modulo 8 arithmetic in adding the constant 1 to the micro-frame number). For any current frame N, if the current micro-frame is 0 to 6, software can safely add isochronous transactions to Frame N + 1. If the current micro-frame is 7, software can add isochronous transactions to Frame N + 2.

Micro-frame caching is indicated with a non-zero value in the least-significant three bits of the isochronous scheduling threshold field. System software assumes the host controller caches one or more periodic data structures for the number of micro-frames indicated in the isochronous scheduling threshold field. For example, if the count value were two, then the host controller keeps a window of two micro-frames worth of state (current micro-frame, plus the next) on-chip. On each micro-frame boundary, the host controller releases the current micro-frame state and begins accumulating the next micro-frame state.

32.6.7 Asynchronous Schedule

The asynchronous schedule traversal is enabled or disabled via the asynchronous schedule enable bit in the USB_USBCMD register. If the asynchronous schedule enable bit is cleared, the host controller simply does not try to access the asynchronous schedule via the USB_ASYNCLISTADDR register. Likewise, if the asynchronous schedule enable bit is set, the host controller does use the USB_ASYNCLISTADDR register to traverse the asynchronous schedule. Modifications to the asynchronous schedule enable bit are not necessarily immediate. The new value of the bit is only taken into consideration the next time the host controller needs to use the value of the USB_ASYNCLISTADDR register to get the next queue head.

The asynchronous schedule status bit in the USB_USBSTS register indicates status of the asynchronous schedule. System software enables (or disables) the asynchronous schedule by writing a one (or zero) to the asynchronous schedule enable bit in the USB_USBCMD register. Software can then poll the asynchronous schedule status bit to determine when the asynchronous schedule has made the desired transition. Software must not modify the asynchronous schedule enable bit unless the value of the asynchronous schedule enable bit equals that of the asynchronous schedule status bit.

The asynchronous schedule manages all control and bulk transfers. Control and bulk transfers are managed using queue head data structures. The asynchronous schedule is based at the USB_ASYNCLISTADDR register. The default value of the USB_ASYNCLISTADDR register after reset is undefined and the schedule is disabled when the asynchronous schedule enable bit is cleared.

Software may only write this register with defined results when the schedule is disabled. For example, asynchronous schedule enable bit in the USB_USBCMD and the asynchronous schedule status bit in the USB_USBSTS register are cleared. System software enables execution from the asynchronous schedule by writing a valid memory address (of a queue head) into this register. Then software enables the asynchronous schedule by setting the asynchronous schedule enable bit is set. The asynchronous schedule is actually enabled when the asynchronous schedule status bit is set.

When the host controller begins servicing the asynchronous schedule, it begins by using the value of the USB_ASYNCLISTADDR register. It reads the first referenced data structure and begins executing transactions and traversing the linked list as appropriate. When the host controller completes processing the asynchronous schedule, it retains the value of the last accessed queue head's horizontal pointer in the USB_ASYNCLISTADDR register. Next time the asynchronous schedule is accessed, this is the first data structure that is serviced. This provides round-robin fairness for processing the asynchronous schedule.

A host controller completes processing the asynchronous schedule when:

- The end of a micro-frame occurs.
- The host controller detects an empty list condition
- The schedule has been disabled via the Asynchronous Schedule Enable bit in the USB_USBCMD register.

The queue heads in the asynchronous list are linked into a simple circular list as shown in Figure 32-52. Queue head data structures are the only valid data structures that may be linked into the asynchronous schedule. An isochronous transfer descriptor (iTD or siTD) in the asynchronous schedule yields undefined results.

The maximum packet size field in a queue head is sized to accommodate the use of this data structure for all non-isochronous transfer types. The USB Specification, Revision 2.0 specifies the maximum packet sizes for all transfer types and transfer speeds. System software should always parameterize the queue head data structures according to the core specification requirements.

32.6.7.1 Adding Queue Heads to Asynchronous Schedule

This is a software requirement section. There are two independent events for adding queue heads to the asynchronous schedule. The first is the initial activation of the asynchronous list. The second is inserting a new queue head into an activated asynchronous list.

Activation of the list is simple. System software writes the physical memory address of a queue head into the USB_ASYNCLISTADDR register, then enables the list by setting the asynchronous schedule enable bit in the USB_USBCMD register to a one.

When inserting a queue head into an active list, software must ensure the schedule is always coherent from the host controllers' point of view. This means the system software must ensure all queue head pointer fields are valid. For example, qTD pointers have T-Bits set or reference valid qTDs and the horizontal pointer references a valid queue head data structure. The following algorithm represents the functional requirements:

```
InsertQueueHead (pQHeadCurrent, pQueueHeadNew)
     --
     -- Requirement: all inputs must be properly initialized.
     --
     -- pQHeadCurrent is a pointer to a queue head that is
     -- already in the active list
     -- pQHeadNew is a pointer to the queue head to be added
     --
     -- This algorithm links a new queue head into an existing
     -- list
     --
     pQueueHeadNew.HorizontalPointer = pQueueHeadCurrent.HorizontalPointer
     pQueueHeadCurrent.HorizontalPointer = physicalAddressOf(pQueueHeadNew)
End InsertQueueHead
```

32.6.7.2 Removing Queue Heads from Asynchronous Schedule

This is a software requirement section. There are two independent events for removing queue heads from the asynchronous schedule. The first is shutting down (deactivating) the asynchronous list. The second is extracting a single queue head from an activated list. Software deactivates the asynchronous schedule by setting the asynchronous schedule enable bit in the USB_USBCMD register to 0. Software can determine when the list is idle when the asynchronous schedule status bit in the USB_USBSTS register is cleared. The normal mode of operation is that software removes queue heads from the asynchronous schedule without shutting it down. Software must not remove an active queue head from the schedule. Software should first deactivate all active qTDs, wait for the queue head to go inactive, then remove the queue head from the asynchronous list. Software removes a queue head from the asynchronous list using the following algorithm. Software merely must ensure all of the link pointers reachable by the host controller are kept consistent.

```
UnlinkQueueHead (pQHeadPrevious, pQueueHeadToUnlink, pQHeadNext)
     --
     -- Requirement: all inputs must be properly initialized.
     --
     -- pQHeadPrevious is a pointer to a queue head that
     -- references the queue head to remove
     -- pQHeadToUnlink is a pointer to the queue head to be
     -- removed
     -- pQheadNext is a pointer to a queue head in the
     -- schedule. Software provides this pointer with the
     -- following strict rules:
     -- if the host software is one queue head, then
     -- pQHeadNext must be the same as
     -- QueueheadToUnlink.HorizontalPointer. If the host
```

```
        -- software is unlinking a consecutive series of
        -- queue heads, QHeadNext must be set by software to
        -- the queue head remaining in the schedule.
        --
        -- This algorithm unlinks a queue head from a circular list
        --
        pQueueHeadPrevious.HorizontalPointer = pQueueHeadToUnlink.HorizontalPointer
        pQueueHeadToUnlink.HorizontalPointer = pQHeadNext
    End UnlinkQueueHead
```

If software removes the queue head with the H-bit set, it must select another queue head linked into the schedule and set its H-bit. This should be completed before removing the queue head. The requirement is that software keep one queue head in the asynchronous schedule, with its H-bit set. At the point software has removed one or more queue heads from the asynchronous schedule, it is unknown whether the host controller has a cached pointer to them. Similarly, it is unknown how long the host controller might retain the cached information, as it is implementation dependent and may be affected by the actual dynamics of the schedule load. Therefore, once software has removed a queue head from the asynchronous list, it must retain the coherency of the queue head (link pointers). It cannot disturb the removed queue heads until it knows that the host controller does not have a local copy of a pointer to any of the removed data structures.

The method software uses to determine when it is safe to modify a removed queue head is to handshake with the host controller. The handshake mechanism allows software to remove items from the asynchronous schedule, then execute a simple, lightweight handshake used by software as a key that it can free (or reuse) the memory associated the data structures it has removed from the asynchronous schedule.

The handshake is implemented with three bits in the host controller. The first bit is a command bit (interrupt on async advance doorbell bit in the USB_USBCMD register) that allows software to inform the host controller that something has been removed from its asynchronous schedule. The second bit is a status bit (interrupt on async advance bit in the USB_USBSTS register) that the host controller sets after it has released all on-chip state that may potentially reference one of the data structures recently removed. When the host controller sets this status bit, it also clears the command bit. The third bit is an interrupt enable (interrupt on async advance bit in the USB_USBINTR register) that is matched with the status bit. If the status bit is set and the interrupt enable bit is set, the host controller asserts a hardware interrupt.

Figure 32-57 illustrates a general example where consecutive queue heads (B and C) are unlinked from the schedule using the algorithm above. Before the unlink operation, the host controller has a copy of queue head A.

The unlink algorithm requires that as software unlinks each queue head, the unlinked queue head is loaded with the address of a queue head that remains in the asynchronous schedule.

When the host controller observes that doorbell bit being set, it makes a note of the local reachable schedule information. In this example, the local reachable schedule information includes both queue heads (A & B). It is sufficient the host controller can set the status bit (and clear the doorbell bit) as soon as it has traversed beyond current reachable schedule information (traversed beyond queue head (B) in this example).

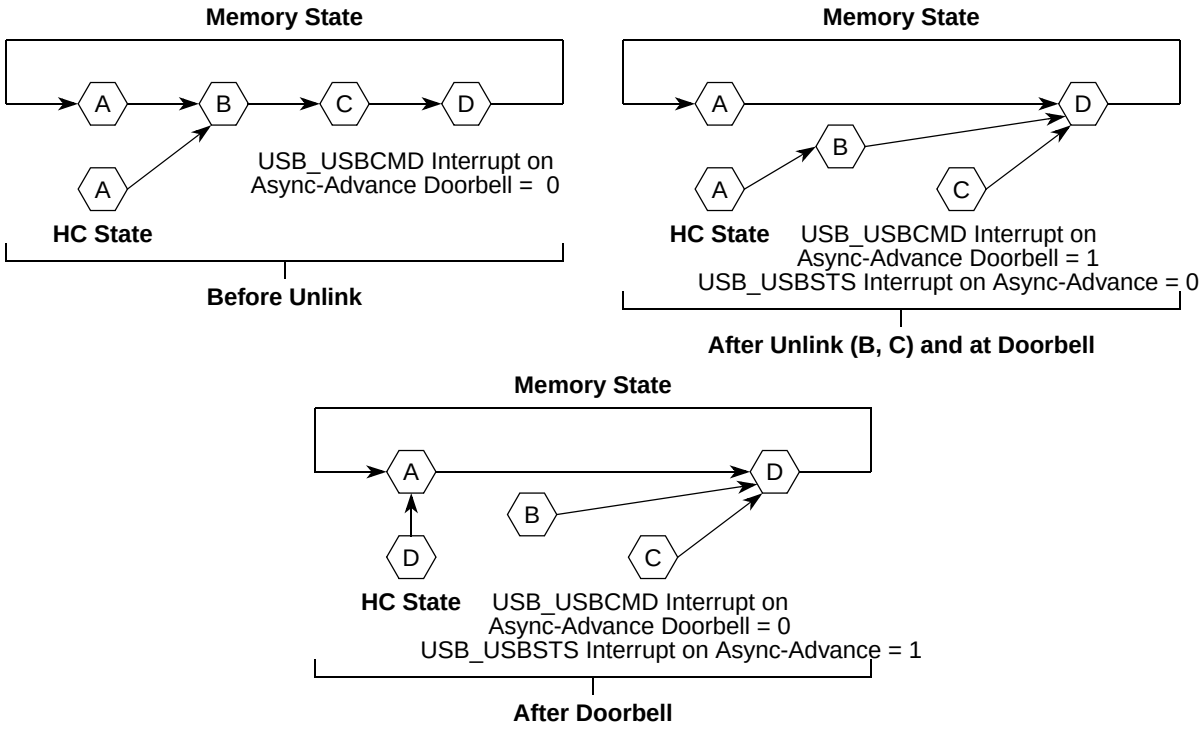


Figure 32-57. Generic Queue Head Unlink Scenario

Alternatively, a host controller implementation is allowed to traverse the entire asynchronous schedule list (for example, observed the head of the queue twice) before setting the advance on async status bit.

Software may re-use the memory associated with the removed queue heads after it observes the interrupt on async advance status bit is set, following assertion of the doorbell. Software should acknowledge the interrupt on async advance status as indicated in the USB_USBSTS register, before using the doorbell handshake again.

32.6.7.3 Empty Asynchronous Schedule Detection

EHCI uses two bits to detect when the asynchronous schedule is empty. The queue head data structure (see Figure 32-49) defines an H-bit in the queue head that allows software to mark a queue head as being the head of the reclaim list. Host controller also keeps a 1-bit flag in the USB_USBSTS register (reclamation) that is cleared when the host controller observes a queue head with the H-bit set. The reclamation flag in the status register is set when any USB transaction from the asynchronous schedule is executed or when the asynchronous schedule starts, see Section 32.6.7.5, “Asynchronous Schedule Traversal: Start Event.”

If the controller ever encounters an H-bit of one and a reclamation bit of zero, the controller simply stops traversal of the asynchronous schedule.

An example illustrating the H-bit in a schedule is shown in Figure 32-58.

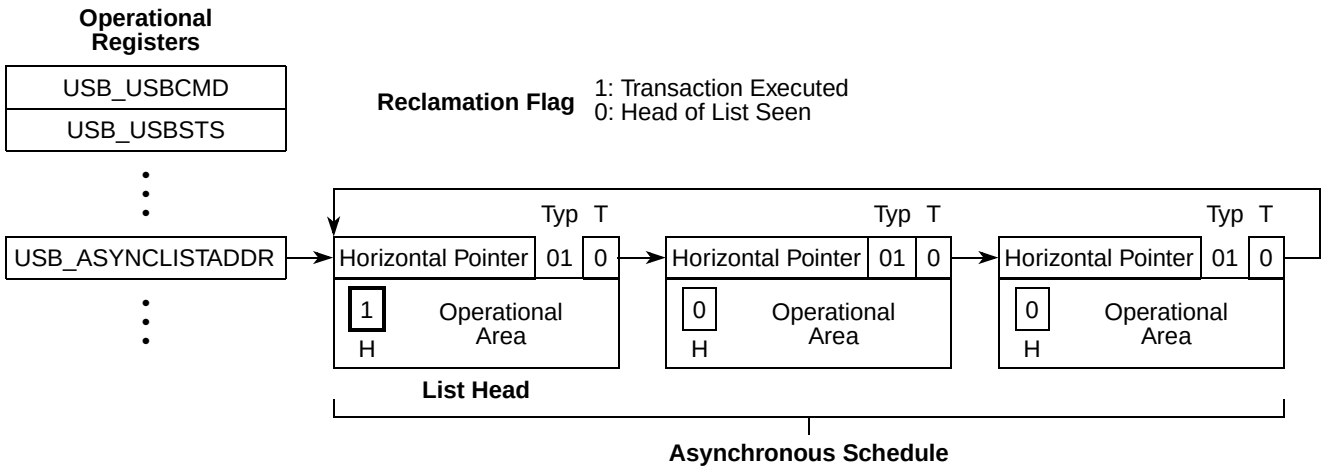


Figure 32-58. Asynchronous Schedule List with Annotation to Mark Head of List

32.6.7.4 Restarting Asynchronous Schedule Before EOF

There are many situations where the host controller detects an empty list before the end of the micro-frame. It is important to remember that under many circumstances the schedule traversal has stopped due to Nak/Nyet responses from all endpoints.

An example of particular interest is when a start-split for a bulk endpoint occurs early in the micro-frame. Given the EHCI simple traversal rules, the complete-split for that transaction may Nak/Nyet out quickly. If it is the only item in the schedule, then the host controller ceases traversal of the Asynchronous schedule early in the micro-frame. To provide reasonable service to this endpoint, the host controller should issue the complete-split before the end of the current micro-frame, instead of waiting until the next micro-frame. When the reason for host controller idling asynchronous schedule traversal is because of empty list detection, it is mandatory the host controller implement a 'waking' method to resume traversal of the asynchronous schedule. An example method is described below.

32.6.7.4.1 Example Method for Restarting Asynchronous Schedule Traversal

The reason for idling the host controller when the list is empty is to keep the host controller from unnecessarily occupying too much memory bandwidth. How long should the host controller stay idle before restarting?

The answer in this example is based on deriving a manifest constant, which is the amount of time the host controller remains idle before restarting traversal. In this example, the manifest constant is called AsyncSchedSleepTime, and has a value of 10 msec. The value is derived based on the analysis in Section 32.6.7.4.2, “Example Derivation for AsyncSchedSleepTime.” The traversal algorithm is simple:

- Traverse the Asynchronous schedule until the either an End-Of-micro-Frame event occurs, or an empty list is detected. If the event is an End-of-micro-Frame, go attempt to traverse the Periodic schedule. If the event is an empty list, then set a sleep timer and go to a schedule sleep state.
- When the sleep timer expires, set working context to the Asynchronous Schedule start condition and go to schedule active state. The start context allows the HC to reload Nakcnt fields, etc. so the HC has a chance to run for more than one iteration through the schedule.

This process simply repeats itself each micro-frame. Figure 32-59 illustrates a sample state machine to manage the active and sleep states of the Asynchronous Schedule traversal policy. There are three states: Actively traversing the Asynchronous schedule, Sleeping, and Not Active. The last two are similar in terms of interaction with the Asynchronous schedule, but the Not Active state means that the host controller is busy with the Periodic schedule or the Asynchronous schedule is not enabled. The Sleeping state is a special state where the host controller is waiting for a period of time before resuming execution of the Asynchronous schedule.

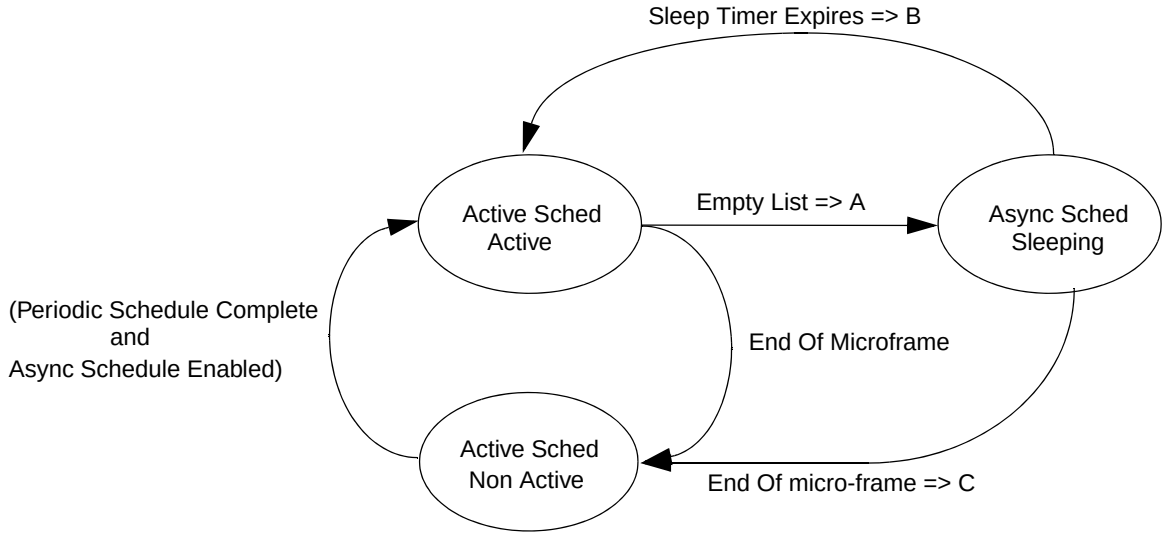


Figure 32-59. Example State Machine for Managing Asynchronous Schedule Traversal

Table 32-71. Asynchronous Schedule State Machine Transition Actions

Action	Action Description Label
A	On detection of the empty list, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.
B	When the *AsynchronousTraversalSleepTimer* expires, the host controller sets the *Reclamation* bit in the USB_USBSTS register to a one and moves the Nak Counter reload state machine to WaitForListHead (see Section 32.6.8, “Operational Model for NAK Counter”).
C	The host controller cancels the sleep timer (*AsynchronousTraversalSleepTimer*).

Async Sched Not Active

This is the initial state of the traversal state machine after a host controller reset. The traversal state machine does not leave this state when the Asynchronous Schedule Enable bit in the USB_USBCMD register is a zero.

This state is entered from Async Sched Active or Async Sched Sleeping states when the end-of-micro-frame event is detected.

Async Sched Active

This state is entered from the Async Sched Not Active state when the periodic schedule is not active. It is also entered from the Async Sched Sleeping states when the AsynchronousTraversalSleepTimer expires. On every transition into this state, the host controller sets the Reclamation bit in the USB_USBSTS register to a one.

While in this state, the host controller continually traverses the asynchronous schedule until either the end of micro-frame or an empty list condition is detected.

Async Sched Sleeping

The state is entered from the Async Sched Active state when a schedule empty condition is detected. On entry to this state, the host controller sets the AsynchronousTraversalSleepTimer to AsyncSchedSleepTime.

32.6.7.4.2 Example Derivation for AsyncSchedSleepTime

The derivation is based on analysis of what work the host controller could be doing next. It assumes the host controller does not keep any state about what work is possibly pending in the asynchronous schedule. The schedule could contain any mix of the possible combinations of high- full- or low-speed control and bulk requests. Table 32-72 summarizes some of the typical “next transactions” that could be in the schedule, and the amount of time (e.g., footprint, or wall clock) the transaction takes to complete.

Table 32-72. Typical Low-/Full Speed Transaction Times

Transaction Attributes	Footprint (time)	Description
Speed: HS Size: 512	11.9 µs	Maximum foot print for a worst case, full sized bulk data.
Speed: HS Size: 512	9.45 µs	Maximum foot print for a best case, full sized bulk data.
Speed: FS Size: 64 Type: Bulk	Approx. 50 µs	Approximate, typical for full sized bulk data.
Speed: FS Size: 8 Type: Control	Approx. 12 µs	Approximate, typical for 8 byte bulk/control (e.g., setup)

An AsyncSchedSleepTime value of 10 ms provides a reasonable relaxation of the system memory load and provides a good level of service for the various transfer types and payload sizes. For example, say you detect an empty list after issuing a start-split for a 64-byte full-speed bulk request. Assuming this is the only thing in the list, the host controller receives the results of the full-speed transaction from the hub during the fifth complete-split request. If the full-speed transaction was an IN and it nak'd, the 10 ms sleep period would allow the host controller to get the NAK results on the first complete-split.

32.6.7.5 Asynchronous Schedule Traversal: Start Event

After the host controller has idled itself using the empty schedule detection, it naturally activates and begins processing from the periodic schedule at the beginning of each micro-frame. In addition, it may have idled itself early in a micro-frame. When this occurs (idles early in the micro-frame), the host controller must occasionally reactivate during the micro-frame and traverse the asynchronous schedule to determine whether any progress can be made. Asynchronous schedule start events are defined to be:

- When the host controller transitions from the periodic schedule to the asynchronous schedule. If the periodic schedule is disabled and the asynchronous schedule is enabled, then the beginning of the micro-frame is equivalent to the transition from the periodic schedule, or
- The asynchronous schedule traversal restarts from a sleeping state.

32.6.7.6 Reclamation Status Bit (USB_USBSTS Register)

The operation of the empty asynchronous schedule detection feature depends on the proper management of the Reclamation bit in the USB_USBSTS register. The host controller tests for an empty schedule after it fetches a new queue head while traversing the asynchronous schedule. The host controller sets the Reclamation bit when an asynchronous schedule traversal start event occurs. The reclamation bit is also set when the host controller executes a transaction while traversing the asynchronous schedule. The host controller clears the reclamation bit when it finds a queue head with its H-bit set. Software should only set a queue head's H-bit if the queue head is in the asynchronous schedule. If software sets the H-bit in an interrupt queue head, the resulting behavior is undefined. The host controller may clear the reclamation bit when executing from the periodic schedule.

32.6.8 Operational Model for NAK Counter

This section describes the operational model for the NakCnt field defined in a queue head (see Section 32.5.6, “Queue Head”). Software should not use this feature for interrupt queue heads. This rule is not required to be enforced by the host controller.

USB protocol has built-in flow control via the NAK response by a device. There are several scenarios, beyond the Ping feature, where an endpoint may naturally NAK or NYET the majority of the time. An example is the host controller management of the split transaction protocol for control and bulk endpoints. All bulk endpoints (High- or Full-speed) are serviced via the same asynchronous schedule. The time between the Start-split transaction and the first Complete-split transaction could be short (e.g., when the endpoint is the only one in the asynchronous schedule). The hub NYETs (effectively NAKs) the Complete-split transaction until the classic transaction is complete. This could result in the host controller thrashing memory, repeatedly fetching the queue head and executing the transaction to the Hub, which does not complete until after the transaction on the classic bus

completes.

There are two component fields in a queue head to support the throttling feature: a counter field (NakCnt), and a counter reload field (RL). NakCnt is used by the host controller as one of the criteria to determine whether or not to execute a transaction to the endpoint. There are two operational modes associated with this counter:

- Not Used. This mode is set when the RL field is zero. The host controller ignores the NakCnt field for any execution of transactions through a queue head with an RL field of zero. Software must use this selection for interrupt endpoints.
- Nak Throttle Mode. This mode is selected when the RL field is non-zero. In this mode, the value in the NakCnt field represents the maximum number of Nak or Nyet responses the host controller tolerates on each endpoint. In this mode, the HC decrements the NakCnt field based on the token/handshake criteria listed in Table 34. The host controller must reload NakCnt when the endpoint successfully moves data (e.g., policy to reward device for moving data).

Table 32-73. NAKCnt Field Adjustment Rules

Token	Handshake NAK	Handshake NYET
IN/PING	Decrement NAKCnt	N/A, Protocol Error
OUT	Decrement NAKCnt	No Action Start
Split	Decrement NAKCnt	N/A, Protocol Error
Complete Split	No Action	Decrement NAKCnt

In summary, system software enables the counter by setting the reload field (RL) to a non-zero value. The host controller may execute a transaction if NakCnt is non-zero. The host controller does not execute a transaction if NakCnt is zero. The reload mechanism is described in detail in Section 32.6.8.1, “NAK Count Reload Control.”

When all queue heads in the Asynchronous Schedule either exhausts all transfers or all NakCnt's go to zero, then the host controller detects an empty Asynchronous Schedule and idle schedule traversal (seeSection 32.6.7.3, “Empty Asynchronous Schedule Detection”).

Any time the host controller begins a new traversal of the Asynchronous Schedule, a Start Event is assumed, see Section 32.6.7.5, “Asynchronous Schedule Traversal: Start Event.” Every time a Start-Event occurs, the Nak Count reload procedure is enabled.

32.6.8.1 NAK Count Reload Control

When the host controller reaches the Execute Transaction state for a queue head (meaning that it has an active operational state), it checks to determine whether the NakCnt field should be reloaded from RL (see Section 32.6.9.3, “Execute Transaction”). If the answer is yes, then RL is copied into NakCnt. After the reload or if the reload is not active, the host controller evaluates whether to execute the transaction.

The host controller must reload nak counters in queue heads (Figure 32-48) during the first pass through the reclamation list after an asynchronous schedule Start Event (see Section 32.6.7.5, “Asynchronous Schedule Traversal: Start Event,” for the definition of the Start Event). The Asynchronous Schedule should have at most one queue head marked as the head (Figure 32-52). Figure 32-60 illustrates an example state machine that satisfies the operational requirements of the host controller detecting the first pass through the Asynchronous Schedule. This state machine is maintained internal to the host controller and is only used to gate reloading of the nak counter during the queue head traversal state: Execute Transaction (Figure 32-61). The host controller does not perform the nak counter reload operation if the RL field (see Figure 32-48) is set to zero.

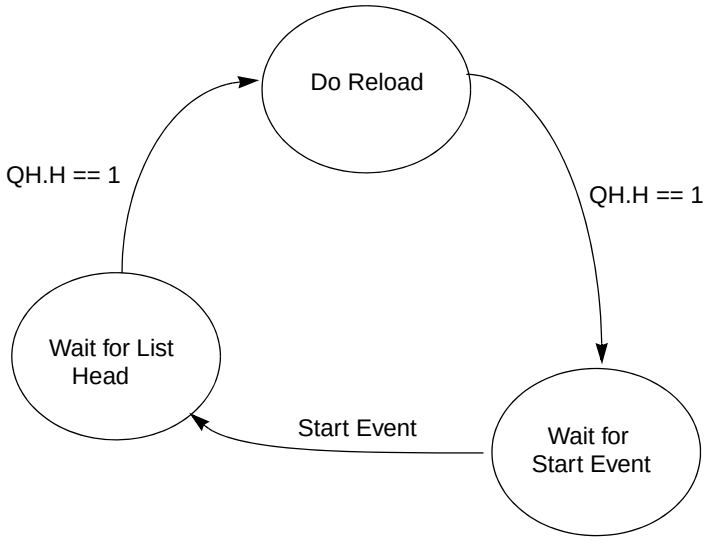


Figure 32-60. Example HC State Machine for Controlling Nak Counter Reloads

32.6.8.1.1 Wait for List Head

This is the initial state. The state machine enters this state from Wait for Start Event when a start event as defined in Section 32.6.7.5, “Asynchronous Schedule Traversal: Start Event,” occurs. The purpose of this state is to wait for the first observation of the head of the Asynchronous Schedule. This occurs when the host controller fetches a queue head whose H-bit is set to a one.

32.6.8.1.2 Do Reload

This state is entered from the Wait for List Head state when the host controller fetches a queue head with the H-bit set to a one. While in this state, the host controller performs nak counter reloads for every queue head visited that has a non-zero nak reload value (RL) field.

32.6.8.1.3 Wait for Start Event

This state is entered from the Do Reload state when a queue head with the H-bit set to a one is fetched. While in this state, the host controller does not perform nak counter reloads.

32.6.9 Managing Control/Bulk/Interrupt Transfers via Queue Heads

This section presents an overview of how the host controller interacts with queuing data structures.

Queue heads use the queue element transfer descriptor (qTD) structure defined in Section 32.5.5, “Queue Element Transfer Descriptor (qTD).”

One queue head is used to manage the data stream for one endpoint. The queue head structure contains static endpoint characteristics and capabilities. It also contains a working area from where individual bus transactions for an endpoint are executed. Each qTD represents one or more bus transactions defined in the context of the EHCI specification as a transfer.

The general processing model for the host controller's use of a queue head is simple:

- Read a queue head
- Execute a transaction from the overlay area
- Write back the results of the transaction to the overlay area
- Move to the next queue head

If the host controller encounters errors during a transaction, the host controller sets one of the error reporting bits in the queue head's status field. The status field accumulates all errors encountered during the execution of a qTD (error bits in the queue head status field are sticky until the transfer [qTD] has completed). This state is always written back to the source qTD when the transfer is complete. On transfer (for example, buffer or halt conditions) boundaries, the host controller must auto-advance (without software intervention) to the next qTD. Additionally, the hardware must be able to halt the queue so no additional bus transactions occur for the endpoint and the host controller does not advance the queue.

An example host controller operational state machine of a queue head traversal is illustrated in Figure 32-61. This state machine is a model for how a host controller should traverse a queue head. The host controller must be able to advance the queue from the Fetch QH state to avoid all hardware/software race conditions. This simple mechanism allows software to simply link qTDs to the queue head and activate them, then the host controller always finds them if/when they are reachable.

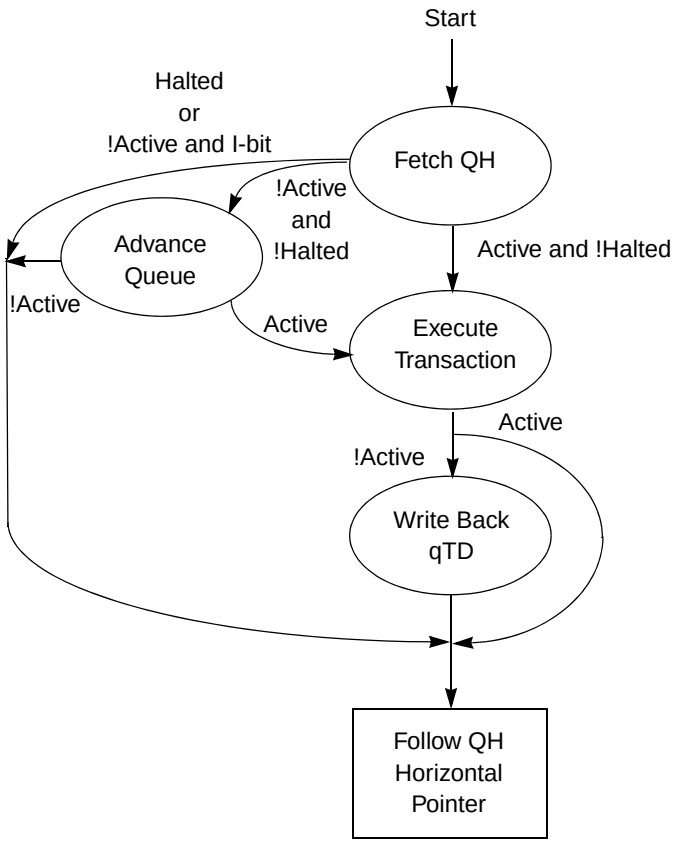


Figure 32-61. Host Controller Queue Head Traversal State Machine

This traversal state machine applies to all queue heads, regardless of transfer type or whether split transactions are required. The following sections describe each state. Each state description describes the entry criteria. The Execute Transaction state (see Section 32.6.9.3, “Execute Transaction”) describes the basic requirements for all endpoints. Section 32.6.11.1, “Split Transactions for Asynchronous Transfers,” and Section 32.6.11.2, “Split Transaction Interrupt,” describe details of the required extensions to the Execute Transaction state for endpoints requiring split transactions.

NOTE

Prior to software placing a queue head into either the periodic or asynchronous list, software must ensure the queue head is properly initialized. Minimally, the queue head should be initialized to the following (see Section 32.5.6, “Queue Head,” for layout of a queue head):

- Valid static endpoint state
- For the first use of a queue head, software may zero out the queue head transfer overlay, then set the *Next qTD Pointer* field value to reference a valid qTD.

32.6.9.1 Fetch Queue Head

A queue head can be referenced from the physical address stored in the USB_ASYNCLISTADDR register (see Section 32.2.4.8, “Current Asynchronous List Address Register (USB_ASYNCLISTADDR)”). Additionally, it may be referenced from the *Next Link Pointer field* of an iTD, siTD, FSTN or another Queue Head. If the referencing link pointer has the *Typ* field set to indicate a queue head, it is assumed to reference a queue head structure as defined in Figure 32-49.

While in this state, the host controller performs operations to implement empty schedule detection (see Section 32.6.7.3, “Empty Asynchronous Schedule Detection”) and Nak Counter reloads (Section 32.6.8, “Operational Model for NAK Counter”). After the queue head has been fetched, the host controller conducts the following queries for empty schedule detection:

- If queue head is not an interrupt queue head (i.e., S-mask is a zero), and
- The H-bit is a one, and
- The Reclamation bit in the USB_USBSTS register is 0.

When these criteria are met, the host controller stops traversing the asynchronous list (as described in Section 32.6.7.3, “Empty Asynchronous Schedule Detection”). When the criteria are not met, the host controller continues schedule traversal. If the queue head is not an interrupt and the H-bit is a one and the Reclamation bit is a one, then the host controller sets the Reclamation bit in the USB_USBSTS register to a zero before completing this state. The operations for reloading of the Nak Counter are described in detail in Section 32.6.8, “Operational Model for NAK Counter.”

This state is complete when the queue head has been read on-chip.

32.6.9.2 Advance Queue

To advance the queue, the host controller must find the next qTD, adjust pointers, perform the overlay and write back the results to the queue head.

This state is entered from the FetchQHD state if the overlay *Active* and *Halt* bits are set to zero. On entry to this state, the host controller determines which next pointer to use to fetch a qTD, fetches a qTD and determines whether or not to perform an overlay. If the I-bit is a one and the Active bit is a zero, the host controller immediately skips processing of this queue head, exits this state and uses the horizontal pointer to the next schedule data structure. If the field Bytes to Transfer is not zero and the T-bit in the Alternate Next qTD Pointer is set to zero, then the host controller uses the Alternate Next qTD Pointer. Otherwise, the host controller uses the Next qTD Pointer. If Next qTD Pointer's T-bit is set to 1, then the host controller exits this state and uses the horizontal pointer to the next schedule data structure.

Using the selected pointer the host controller fetches the referenced qTD. If the fetched qTD has its Active bit set to a one, the host controller moves the pointer value used to reach the qTD (Next or Alternate Next) to the Current qTD Pointer field, then performs the overlay. If the fetched qTD has its Active bit set to a zero, the host controller aborts the queue advance and follows the queue head's horizontal pointer to the next schedule data structure. The host controller performs the overlay based on the following rules:

- The value of the data toggle (dt) field in the overlay area depends on the value of the data toggle control (dtc) bit.
- If the EPS field indicates the endpoint is a high-speed endpoint, the Ping state field is preserved by the host controller. The value of this field is not changed as a result of the overlay.
- C-prog-mask field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- Frame Tag field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- NakCnt field in the overlay area is loaded from the RL field in the queue head's Static Endpoint State.
- All other areas of the overlay are set by the incoming qTD.

The host controller exits this state when it has committed the write to the queue head.

32.6.9.3 Execute Transaction

The host controller enters this state from the Fetch Queue Head state only if the *Active* bit in Status field of the queue head is set to a one.

On entry to this state, the host controller executes a few pre-operations, then checks some pre-condition criteria before committing to executing a transaction for the queue head.

The pre-operations performed and the pre-condition criteria depend on whether the queue head is an interrupt endpoint. The host controller can determine that a queue head is an interrupt queue head when the queue head's S-mask field contains a non-zero value. It is the responsibility of software to ensure the S-mask field is appropriately initialized based on the transfer type. There are other criteria that must be met if the EPS field indicates that the endpoint is a low- or full-speed endpoint, see Section 32.6.11.1, "Split Transactions for Asynchronous Transfers," and Section 32.6.11.2, "Split Transaction Interrupt."

- Interrupt Transfer Pre-condition Criteria

  If the queue head is for an interrupt endpoint (non-zero *S-mask* field), then the USB_FRINDEX[2:0] field must identify a bit in the S-mask field that has a one in it. For example,

an S-mask value of 00100000b would evaluate to true only when USB_FRINDEX[2:0] is equal to 101b. If this condition is met then the host controller considers this queue head for a transaction.

- Asynchronous Transfer Pre-operations and Pre-condition Criteria

  If the queue head is not for an interrupt endpoint (a zero S-mask field), then the host controller performs one pre-operation and then evaluates one pre-condition criteria: The pre-operation is:

  — Checks the Nak counter reload state. It may be necessary for the host controller to reload the Nak Counter field. The reload is performed at this time.

  The pre-condition evaluated is:

  — Whether or not the NakCnt field has been reloaded, the host controller checks the value of the NakCnt field in the queue head. If NakCnt is non-zero, or if the Reload Nak Counter field is zero, then the host controller considers this queue head for a transaction.

- Transfer Type Independent Pre-operations

  Regardless of the transfer type, the host controller always performs at least one pre-operation and evaluates one pre-condition. The pre-operation is:

  — A host controller internal transaction (down) counter qHTransactionCounter is loaded from the queue head's Mult field. A host controller implementation is allowed to ignore this for queue heads on the asynchronous list. It is mandatory for interrupt queue heads. Software should ensure that the Mult field is set appropriately for the transfer type.

  The pre-conditions evaluated are:

  — The host controller determines whether there is enough time in the micro-frame to complete this transaction. If there is not enough time to complete the transaction, the host controller exits this state.

  — If the value of qHTransactionCounter for an interrupt endpoint is zero, then the host controller exits this state.

When the pre-operations are complete and pre-conditions are met, the host controller sets the Reclamation bit in the USB_USBSTS register to a one and then begins executing one or more transactions using the endpoint information in the queue head. The host controller iterates qHTransactionCounter times in this state executing transactions. After each transaction is executed, qHTransactionCounter is decremented by one. The host controller exits this state when one of the following events occurs:

- The qHTransactionCounter decrements to zero, or
- The endpoint responds to the transaction with any handshake other than an ACK, or
- The transaction experiences a transaction error, or
- The Active bit in the queue head goes to a zero, or
- There is not enough time in the micro-frame left to execute the next transaction.

NOTE

For a high-bandwidth interrupt OUT endpoint, the host controller may optionally immediately retry the transaction if it fails.

The results of each transaction is recorded in the on-chip overlay area. If data was successfully moved during the transaction, the transfer state in the overlay area is advanced. To advance the queue head's transfer state, the Total Bytes to Transfer field is decremented by the number of bytes moved in the

transaction, the data toggle bit (dt) is toggled, the current page offset is advanced to the next appropriate value (e.g., advanced by the number of bytes successfully moved), and the C_Page field is updated to the appropriate value (if necessary). See Section 32.6.9.6, “Buffer Pointer List Use for Data Streaming with qTDs.”

The total bytes to transfer field may be zero when all the other criteria for executing a transaction are met. When this occurs, the host controller executes a zero-length transaction to the endpoint. If the PID_Code field indicates an IN transaction and the device delivers data, the host controller detects a packet babble condition, set the babble and halted bits in the Status field, set the Active bit to a zero, write back the results to the source qTD, then exit this state.

In the event an IN token receives a data PID mismatch response, the host controller must ignore the received data (i.e., not advance the transfer state for the bytes received). Additionally, if the endpoint is an interrupt IN, then the host controller must record that the transaction occurred (i.e., decrement qHTransactionCounter). It is recommended (but not required) the host controller continue executing transactions for this endpoint if the resultant value of *qHTransactionCounter* is greater than one.

If the response to the IN bus transaction is a Nak (or Nyet) and RL is non-zero, NakCnt is decremented by one. If RL is zero, then no write-back by the host controller is required (for a transaction receiving a Nak or Nyet response and the value of CErr did not change). Software should set the RL field to zero if the queue head is an interrupt endpoint. Host controller hardware is not required to enforce this rule or operation.

After the transaction has finished and the host controller has completed the post processing of the results (advancing the transfer state and possibly NakCnt, the host controller writes back the results of the transaction to the queue head’s overlay area in main memory.

The number of bytes moved during an IN transaction depends on how much data the device endpoint delivers. The maximum number of bytes a device can send is Maximum Packet Size. The number of bytes moved during an OUT transaction is either Maximum Packet Length bytes or Total Bytes to Transfer, whichever is less.

If there was a transaction error during the transaction, the transfer state (as defined above) is not advanced by the host controller. The CErr field is decremented by one and the status field is updated to reflect the type of error observed. Transaction errors are summarized in Section 32.6.14.1.1, “Transaction Error.”

The following events causes the host controller to clear the Active bit in the queue head’s overlay status field. When the Active bit transitions from a 1 to a 0, the transfer in the overlay is considered complete. The reason for the transfer completion (clearing the Active bit) determines the next state.

- CErr field decrements to 0. When this occurs the Halted bit is set to a one and Active is set to a zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The device responds to the transaction with a STALL PID. When this occurs, the Halted bit is set to a one and the Active bit is set to a zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The Total Bytes to Transfer field is zero after the transaction completes. For a zero length transaction, it was zero before the transaction was started. When this condition occurs, the Active bit is set to zero.

- The PID code is an IN, and the number of bytes moved during the transaction is less than the Maximum Packet Length. When this occurs, the Active bit is set to zero and a short packet condition exists. The short-packet condition is detected during the Advance Queue state. Refer to Section 5.12 for additional rules for managing low- and full-speed transactions.
- The PID Code field indicates an IN and the device sends more than the expected number of bytes (e.g., Maximum Packet Length or Total Bytes to Transfer bytes, whichever is less) (i.e., a packet babble). This results in the host controller setting the Halted bit to a one.

With the exception of a NAK response (when RL field is zero), the host controller always writes the results of the transaction back to the overlay area in main memory. This includes when the transfer completes. For a high-speed endpoint, the queue head information written back includes minimally the following fields:

- NakCnt, dt, Total Bytes to Transfer, C_Page, Status, CERR, and Current Offset

For a low- or full-speed device the queue head information written back also includes the fields:

- C-prog-mask, FrameTag and S-bytes.

The duration of this state depends on the time it takes to complete the transaction(s) and the status write to the overlay is committed.

32.6.9.3.1 Halting a Queue Head

A halted endpoint is defined only for the transfer types that are managed via queue heads (control, bulk and interrupt). The following events indicate that the endpoint has reached a condition where no more activity can occur without intervention from the driver:

- An endpoint may return a STALL handshake during a transaction,
- A transaction had three consecutive error conditions, or
- A Packet Babble error occurs on the endpoint.

When any of these events occur (for a queue head) the Host Controller halts the queue head and set the USBERRINT status bit in the USB_USBSTS register to a one. To halt the queue head, the Active bit is set to a zero and the Halted bit is set to a one. There may be other error status bits that are set when a queue is halted. The host controller always writes back the overlay area to the source qTD when the transfer is complete, regardless of the reason (normal completion, short packet or halt). The host controller does not advance the transfer state on a transaction that results in a *Halt* condition (i.e., no updates necessary for Total Bytes to Transfer, C_Page, Current Offset, and dt). The host controller must update CErr as appropriate. When a queue head is halted, the USB Error Interrupt bit in the USB_USBSTS register is set to a one. If the USB Error Interrupt Enable bit in the USB_USBINTR register is set to a one, a hardware interrupt is generated at the next interrupt threshold.

32.6.9.3.2 Asynchronous Schedule Park Mode

Asynchronous Schedule Park mode is a special execution mode that can be enabled by system software, where the host controller is permitted to execute more than one bus transaction from a high-speed queue head in the Asynchronous schedule before continuing horizontal traversal of the Asynchronous schedule. This feature has no effect on queue heads or other data structures in the Periodic schedule. This feature is similar in intent to the Mult feature that is used in the Periodic schedule. Whereas the Mult feature is a

characteristic that is tunable for each endpoint, park-mode is a policy that is applied to all high-speed queue heads in the asynchronous schedule. It is essentially the specification of an iterator for consecutive bus transactions to the same endpoint. All of the rules for managing bus transactions and the results of those as defined in Section 32.6.9.3, “Execute Transaction,” apply. This feature merely specifies how many consecutive times the host controller is permitted to execute from the same queue head before moving to the next queue head in the Asynchronous List. This feature should allow the host controller to attain better bus utilization for those devices that are capable of moving data at maximum rate, while at the same time providing a fair service to all endpoints.

A host controller exports its capability to support this feature to system software by setting the Asynchronous Schedule Park Capability bit in the USB_HCCPARAMS register to 1. This information keys system software that the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USB_USBCMD register are modifiable. System software enables the feature by writing a one to the Asynchronous Schedule Park Mode Enable bit.

When park-mode is not enabled (Asynchronous Schedule Park Mode Enable bit in the USB_USBCMD register is 0), the host controller must not execute more than one bus transaction per high-speed queue head, per traversal of the asynchronous schedule. When park-mode is enabled, the host controller must not apply the feature to a queue head whose EPS field indicates a Low/Full-speed device (i.e., only one bus transaction is allowed from each Low/Full-speed queue head per traversal of the asynchronous schedule). Park-mode may only be applied to queue heads in the Asynchronous schedule whose EPS field indicates that it is a high-speed device.

The host controller must apply park mode to queue heads whose EPS field indicates a high-speed endpoint. The maximum number of consecutive bus transactions a host controller may execute on a high-speed queue head is determined by the value in the Asynchronous Schedule Park Mode Count field in the USB_USBCMD register. Software must not set Asynchronous Schedule Park Mode Enable bit to a one and also set Asynchronous Schedule Park Mode Count field to a zero. The resulting behavior is not defined. An example behavioral example describes the operational requirements for the host controller implementing park-mode. This feature does not affect how the host controller handles the bus transaction as defined in Section 32.6.9.3, “Execute Transaction.” It only affects how many consecutive bus transactions for the current queue head can be executed. All boundary conditions, error detection and reporting applies as usual. This feature is similar in concept to the use of the Mult field for high-bandwidth Interrupt for queue heads in the Periodic Schedule.

The host controller effectively loads an internal down-counter PM-Count from Asynchronous Schedule Park Mode Count when Asynchronous Schedule Park Mode Enable bit is a one, and a high-speed queue head is first fetched and meets all the criteria for executing a bus transaction. After the bus transaction, PM-Count is decremented. The host controller may continue to execute bus transactions from the current queue head until PM-Count goes to zero, an error is detected, the buffer for the current transfer is exhausted or the endpoint responds with a flow-control or STALL handshake.

32.6.9.4 Write Back qTD

This state is entered from the Execute Transaction state when the *Active* bit is set to a zero. The source data for the write-back is the transfer results area of the queue head overlay area (see Figure 32-49). The host controller uses the Current qTD Pointer field as the target address for the qTD. The queue head transfer

result area is written back to the transfer result area of the target qTD. This state is also referred to as: qTD retirement. The fields that must be written back to the source qTD include Total Bytes to Transfer, Cerr, and Status.

The duration of this state depends on when the qTD write-back has been committed.

32.6.9.5 Follow Queue Head Horizontal Pointer

The host controller must use the horizontal pointer in the queue head to the next schedule data structure when any of the following conditions exist:

- If the Active bit is a one on exit from the Execute Transaction state, or
- When the host controller exits the Write Back qTD state, or
- If the Advance Queue state fails to advance the queue because the target qTD is not active, or
- If the Halted bit is a one on exit from the Fetch QH state.

There is no functional requirement that the host controller wait until the current transaction is complete before using the horizontal pointer to read the next linked data structure. However, it must wait until the current transaction is complete before executing the next data structure.

32.6.9.6 Buffer Pointer List Use for Data Streaming with qTDs

A qTD has an array of buffer pointers used to reference the data buffer for a transfer. The EHCI specification requires the buffer associated with the transfer be virtually contiguous. If the buffer spans more than one physical page, it must obey the following rules:

- The first portion of the buffer must begin at some offset in a page and extend through the end of the page.
- The remaining buffer cannot be allocated in small chunks scattered around memory. For each 4K chunk beyond the first page, each buffer portion matches to a full 4K page. The final portion, which may only be large enough to occupy a portion of a page, must start at the top of the page and be contiguous within that page.

Figure 32-62 illustrates these requirements.

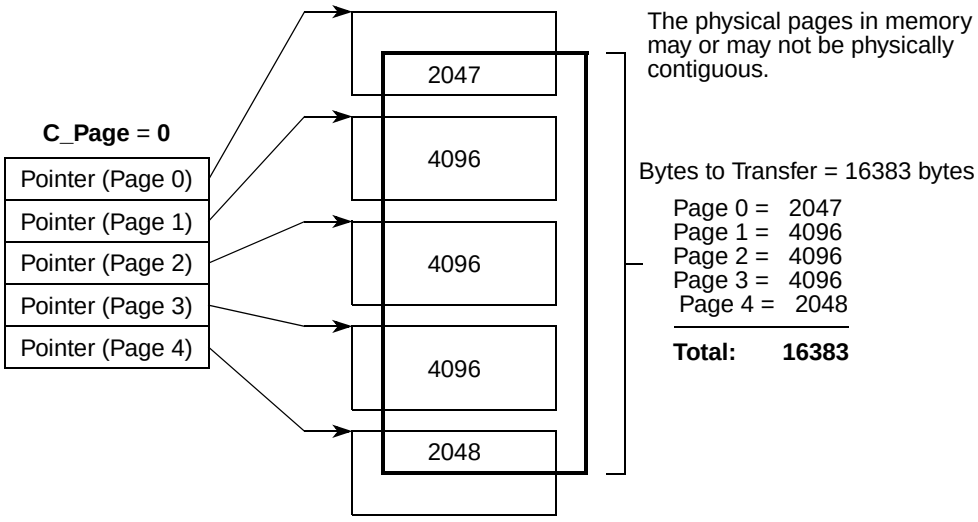


Figure 32-62. Example Mapping of qTD Buffer Pointers to Buffer Pages

The buffer pointer list in the qTD is long enough to support a maximum transfer size of 20K bytes. This case occurs when all five buffer pointers are used and the first offset is zero. A qTD manages a 16 KB buffer with any starting buffer alignment.

The host controller uses the C_Page field as an index value to determine which buffer pointer in the list should be used to start the current transaction. The host controller uses a different buffer pointer for each physical page of the buffer. This is true even if the buffer is physically contiguous.

The host controller must detect when the current transaction spans a page boundary and automatically moves to the next available buffer pointer in the page pointer list. The next available pointer is reached by incrementing C_Page and pulling the next page pointer from the list. Software must ensure there are sufficient buffer pointers to move the amount of data specified in the bytes to transfer field.

Figure 32-62 illustrates a nominal example of how system software would initialize the buffer pointers list and the C_Page field for a transfer size of 16383 bytes. C_Page is cleared. The upper 20-bits of Page 0 references the start of the physical page. Current offset (the lower 12-bits of queue head doubleword 7) holds the offset in the page for example, 2049 (for example, 4096-2047). The remaining page pointers are set to reference the beginning of each subsequent 4K page.

For the first transaction on the qTD (assuming a 512-byte transaction), the host controller uses the first buffer pointer (page 0 because C_Page is cleared) and concatenates the current offset field. The 512 bytes are moved during the transaction, and the current offset and total bytes to transfer are adjusted by 512 and written back to the queue head working area.

During the fourth transaction, the host controller needs 511 bytes in page 0 and one byte in page 1. The host controller increments C_Page (to 1) and uses the page 1 pointer to move the final byte of the transaction. After the fourth transaction, the active page pointer is the page 1 pointer and current offset has rolled to one, and both are written back to the overlay area. The transactions continue for the rest of the buffer, with the host controller automatically moving to the next page pointer (that is, C_Page) when necessary. There are three conditions for how the host controller manages C_Page.

- The current transaction does not span a page boundary. The value of C_Page is not adjusted by the host controller.
- The current transaction does span a page boundary. The host controller must detect the page cross condition and advance to the next buffer while streaming data to/from the USB.
- The current transaction completes on a page boundary (the last byte moved for the current transaction is the last byte in the page for the current page pointer). The host controller must increment C_Page before writing back status for the transaction.

The only valid adjustment the host controller may make to C_Page is to increment by one.

32.6.9.7 Adding Interrupt Queue Heads to the Periodic Schedule

The link path(s) from the periodic frame list to a queue head establishes in which frames a transaction can be executed for the queue head. Queue heads are linked into the periodic schedule so they are polled at the appropriate rate. System software sets a bit in a queue head's S-Mask to indicate which micro-frame within a one millisecond period a transaction should be executed for the queue head. Software must ensure all queue heads in the periodic schedule have S-Mask set to a non-zero value. An S-mask with a zero value in the context of the periodic schedule yields undefined results.

If the desired poll rate is greater than one frame, system software can use a combination of queue head linking and S-Mask values to spread interrupts of equal poll rates through the schedule so that the periodic bandwidth is allocated and managed in the most efficient manner possible. Some examples are illustrated in Table 32-74.

Table 32-74. Example Periodic Reference Patterns for Interrupt Transfers

Frame Reference Sequence	Description
0, 2, 4, 6, 8,.... S-Mask = 0x01	A queue head for the bInterval of two milliseconds (16 micro-frames) is linked into the periodic schedule so it is reachable from the periodic frame list locations indicated in the previous column. In addition, the S-Mask field in the queue head is set to 0x01, indicating that the transaction for the endpoint should be executed on the bus during micro-frame 0 of the frame.
0, 2, 4, 6, 8,... S-Mask = 0x02	Another example of a queue head with a bInterval of two milliseconds is linked into the periodic frame list at exactly the same interval as the previous example. However, the S-Mask is set to 0x02 indicating that the transaction for the endpoint should be executed on the bus during micro-frame 1 of the frame.

32.6.9.8 Managing Transfer Complete Interrupts from Queue Heads

The host controller sets an interrupt to be signaled at the next interrupt threshold when the completed transfer (qTD) has an interrupt on complete (IOC) bit set, or when a transfer (qTD) completes with a short packet. If system software needs multiple qTDs to complete a client request (like a control transfer) the intermediate qTDs do not require interrupts. System software may only need a single interrupt to notify it that the complete buffer has been transferred. System software may set IOCs to occur more frequently. A motivation for this may be that it wants early notification so interface data structures can be re-used in a timely manner.

32.6.10 Ping Control

USB 2.0 defines an addition to the protocol for high-speed devices called ping. Ping is required for all USB 2.0 high-speed bulk and control endpoints. Ping is not allowed for a split-transaction stream. This extension to the protocol eliminates the bad side effects of NAKing OUT endpoints. The status field has a ping state bit that the host controller uses to determine the next actual PID it uses in the next transaction to the endpoint (see Table 32-59). The ping state bit is only managed by the host controller for queue heads that meet all of the following criteria:

- The queue head is not an interrupt
- The EPS field equals High-Speed
- The PIDCode field equals OUT

Table 32-75 illustrates the state transition table for the host controller's responsibility for maintaining the PING protocol. Refer to Chapter 8 in the *USB Specification, Revision 2.0* for detailed description on the Ping protocol.

Table 32-75. Ping Control State Transition Table

	Event		
Current	**Host**	**Device**	**Next**
Do Ping	PING	Nak	Do Ping
Do Ping	PING	Ack	Do OUT
Do Ping	PING	XactErr[1]	Do Ping
Do Ping	PING	Stall	N/C[2]
Do OUT	OUT	Nak	Do Ping
Do OUT	OUT	Nyet	Do Ping[3]
Do OUT	OUT	Ack	Do OUT
Do OUT	OUT	XactErr[1]	Do Ping
Do OUT	OUT	Stall	N/C[2]

1 Transaction Error (XactErr) is any time the host misses the handshake.

2 No transition change required for the ping state bit. The stall handshake results in the endpoint being halted (for example, active cleared and halt set). Software intervention is required to restart queue.

3 A Nyet response to an OUT means that the device has accepted the data, but cannot receive any more at this time. Host must advance the transfer state and transition the ping state bit to do ping.

The ping state bit is described in Table 32-59. The defined ping protocol allows the host to be imprecise on the initialization of the ping protocol (start in Do OUT when you don't know whether there is space on the device or not). The host controller manages the ping state bit. System software sets the initial value in the queue head when it initializes a queue head. The host controller preserves the ping state bit across all queue advancements. This means when a new qTD is written into the queue head overlay area, the previous value of the ping state bit is preserved.

32.6.11 Split Transactions

USB 2.0 defines extensions to the bus protocol for managing USB 1.x data streams through USB 2.0 hubs. This section describes how the host controller uses the interface data structures to manage data streams with full- and low-speed devices, connected below a USB 2.0 hub, utilizing the split transaction protocol. Refer to the USB 2.0 Specification for the complete definition of the split transaction protocol. Full- and low-speed devices are enumerated identically as high-speed devices, but the transactions to the full- and low-speed endpoints use the split-transaction protocol on the high-speed bus. The split transaction protocol is an encapsulation of (or wrapper around) the full- or low-speed transaction. The high-speed wrapper portion of the protocol is addressed to the USB 2.0 hub and transaction translator below that the full- or low-speed device is attached.

EHCI uses dedicated data structures for managing full-speed isochronous data streams. Control, bulk, and interrupt are managed using the queuing data structures. The interface data structures need to be programmed with the device address and the transaction translator number of the USB 2.0 hub operating as the low-/full-speed host controller for this link. The following sections describe how the host controller processes and manages the split transaction protocol.

32.6.11.1 Split Transactions for Asynchronous Transfers

A queue head in the asynchronous schedule with an EPS field indicating a full-or low-speed device indicates to the host controller that it must use split transactions to stream data for this queue head. All full-speed bulk and full- and low-speed control are managed via queue heads in the asynchronous schedule.

Software must initialize the queue head with the appropriate device address and port number for the transaction translator serving as the full-/low-speed host controller for the links connecting the endpoint. Software must also initialize the split transaction state bit (SplitXState) to Do-Start-Split. Finally, if the endpoint is a control endpoint, system software must set the control transfer type (C) bit in the queue head to a one. If this is not a control transfer type endpoint, the C bit must be initialized by software to be a zero. The host controller uses this information to properly set the endpoint type (ET) field in the split transaction bus token. When the C bit is a zero, the split transaction token's ET field is set to indicate a bulk endpoint. When the C bit is a one, the split transaction token's ET field is set to indicate a control endpoint. Refer to Chapter 8 of *USB Specification, Revision 2.0* for details.

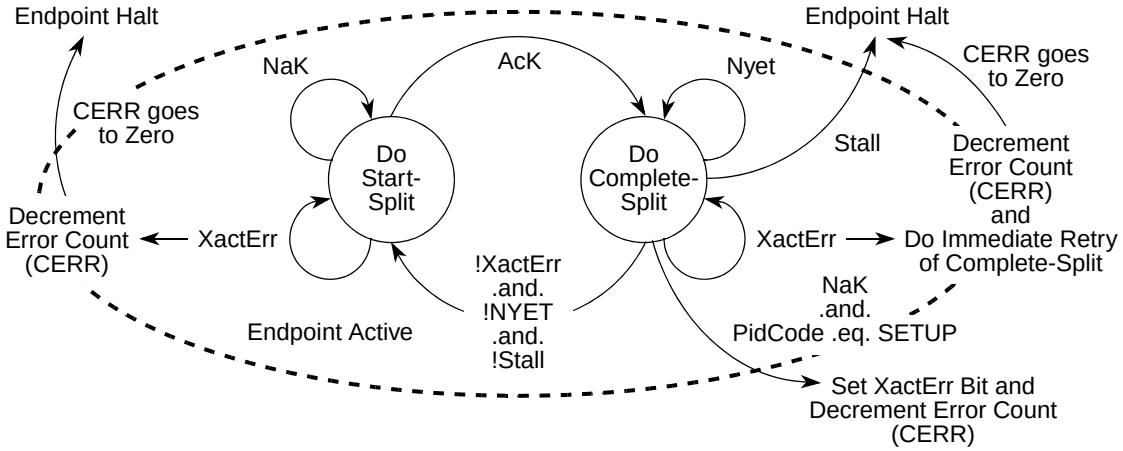


Figure 32-63. Host Controller Asynchronous Schedule Split-Transaction State Machine

32.6.11.1.1 Asynchronous—Do-Start-Split

Do-Start-Split is the state that software must initialize a full- or low-speed asynchronous queue head. This state is entered from the Do-Complete-Split state only after a complete-split transaction receives a valid response from the transaction translator that is not a Nyet handshake.

For queue heads in this state, the host controller executes a start-split transaction to the transaction translator. If the bus transaction completes without an error and PID code indicates an IN or OUT transaction, then the host controller reloads the error counter (Cerr). If it is a successful bus transaction and the PID Code indicates a SETUP, the host controller does not reload the error counter. If the transaction translator responds with a NAK, the queue head is left in this state, and the host controller proceeds to the next queue head in the asynchronous schedule.

If the host controller times out the transaction (no response, or bad response), the host controller decrements Cerr and proceeds to the next queue head in the asynchronous schedule.

32.6.11.1.2 Asynchronous—Do-Complete-Split

This state is entered from the Do-Start-Split state only after a start-split transaction receives an ACK handshake from the transaction translator.

For queue heads in this state, the host controller executes a complete-split transaction to the transaction translator. If the transaction translator responds with a Nyet handshake, the queue head is left in this state, the error counter is reset, and the host controller proceeds to the next queue head in the asynchronous schedule. When a Nyet handshake is received for a bus transaction where the queue head's PID code indicates an IN or OUT, the host controller reloads the error counter (Cerr). When a Nyet handshake is received for a complete-split bus transaction where the queue head's PID Code indicates a SETUP, the host controller must not adjust the value of Cerr.

Independent of PID code, the following responses have the indicated effects:

- Transaction Error (XactErr). Timeout/data CRC failure. The error counter (Cerr) is decremented by one and the complete split transaction is immediately retried (if possible). If there is not enough time in the micro-frame to execute the retry, the host controller ensures that the next time the host

controller begins executing from the asynchronous schedule, it must begin executing from this queue head. If another start-split (for some other endpoint) is sent to the transaction translator before the complete-split is really completed, the transaction translator could dump the results (which were never delivered to the host). This is why the core specification states the retries must be immediate. When the host controller returns to the asynchronous schedule in the next micro-frame, the first transaction from the schedule is the retry for this endpoint. If Cerr went to zero, the host controller halts the queue.

- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced and the state is exited. If the PID Code is a SETUP, then the Nak response is a protocol error. The XactErr status bit is set and the Cerr field is decremented.
- STALL. The target endpoint responded with a STALL handshake. The host controller sets the halt bit in the status byte and retires the qTD, but does not attempt to advance the queue.

If the PID code indicates an IN, then any of following responses are expected:

- DATA0/1. On reception of data, the host controller ensures the PID matches the expected data toggle and checks CRC. If the packet is good, the host controller advances the state of the transfer (for example, moves the data pointer by the number of bytes received, decrements the BytesToTransfer field by the number of bytes received, and toggles the dt bit). The host controller then exits this state. The response and advancement of transfer may trigger other processing events, such as retirement of the qTD and advancement of the queue.

  If the data sequence PID does not match the expected, the data is ignored, the transfer state is not advanced, and this state is exited.

If the PID code indicates an OUT/SETUP, any of following responses are expected:

- ACK. The target endpoint accepted the data, so the host controller must advance the state of the transfer. The current offset field is incremented by maximum packet length or bytes to transfer, whichever is less. The bytes to transfer field is decremented by the same amount and the data toggle bit (dt) is toggled. The host controller then exits this state.

  Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue.

32.6.11.2 Split Transaction Interrupt

Split-transaction interrupt-IN/OUT endpoints are managed using the same data structures used for high-speed interrupt endpoints. They both co-exist in the periodic schedule. Queue heads/qTDs offer the set of features required for reliable data delivery, which is characteristic to interrupt transfer types. The split-transaction protocol is managed completely within this defined functional transfer framework. For example, the host controller visits a queue head, executes a high-speed transaction (if criteria are met), and advances the transfer state (or not) depending on the results of the entire transaction for a high-speed endpoint. For low- and full-speed endpoints, the details of the execution phase are different (that is, takes more than one bus transaction to complete), but the remainder of the operational framework is intact.

32.6.11.2.1 Split Transaction Scheduling Mechanisms for Interrupt

Full- and low-speed interrupt queue heads have an EPS field indicating full- or low-speed and have a non-zero S-mask field. The host controller can detect this combination of parameters and assume the endpoint is a periodic endpoint. Low- and full-speed interrupt queue heads require the use of the split transaction protocol. The host controller sets the endpoint type (ET) field in the split token to indicate the transaction is an interrupt. These transactions are managed through a transaction translator's periodic pipeline. Software should not set these fields to indicate the queue head is an interrupt unless the queue head is used in the periodic schedule.

System software manages the per/transaction translator periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each endpoint occur. The characteristics of the transaction translator are such that the high-speed transaction protocol must execute during explicit micro-frames or the data or response information in the pipeline is lost. Figure 32-64 illustrates the general scheduling boundary conditions supported by the EHCI periodic schedule and queue head data structure. The S and C_n labels indicate micro-frames where software can schedule start-splits and complete splits (respectively).

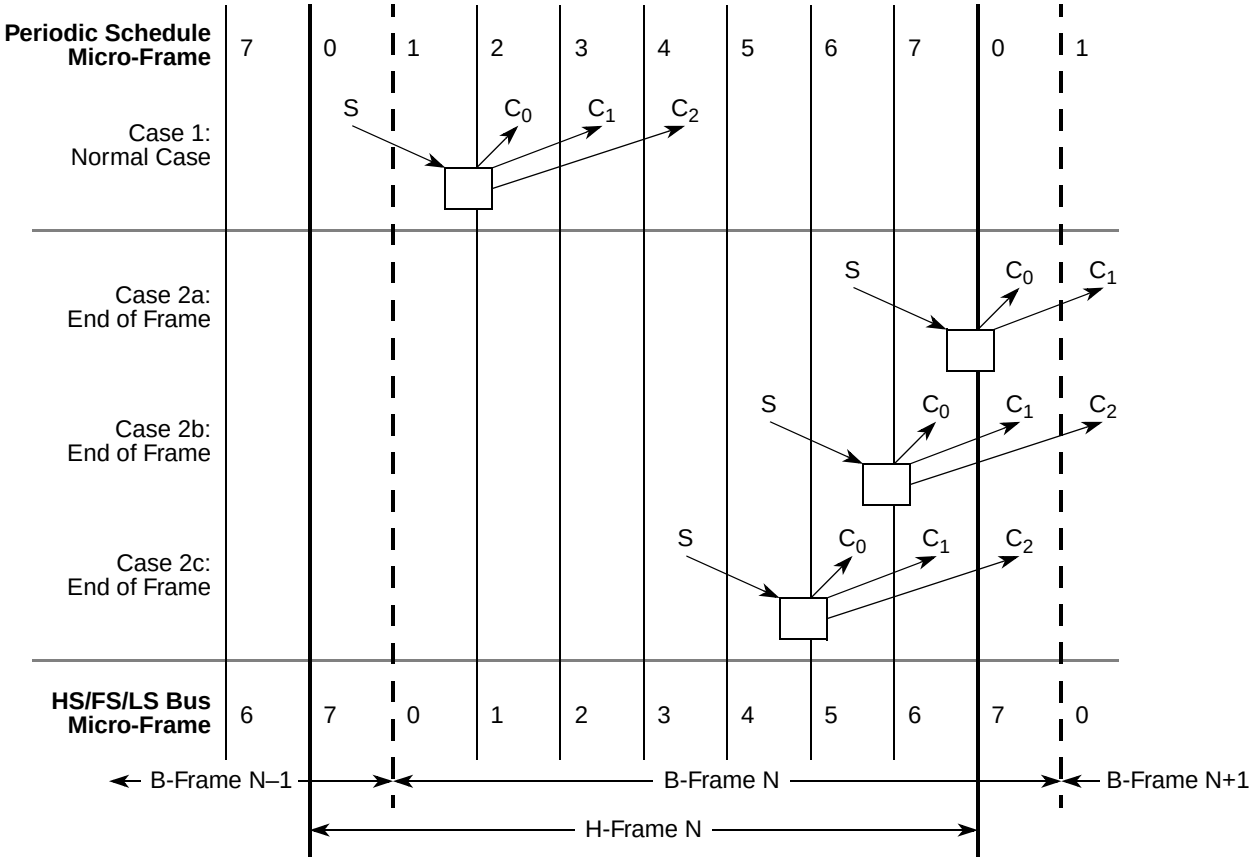


Figure 32-64. Split Transaction and Interrupt Scheduling Boundary Conditions

The scheduling cases are:

- Case 1: The normal scheduling case is where the entire split transaction is completely bounded by a frame (H-Frame in this case).

- Case 2a through Case 2c: The USB 2.0 hub pipeline rules states clearly, when and how many complete-splits must be scheduled to account for earliest to latest execution on the full/low-speed link. The complete-splits may span the H-Frame boundary when the start-split is in micro-frame 4 or later. When this occurs, the H-Frame to B-Frame alignment requires the queue head be reachable from consecutive periodic frame list locations. System software cannot build an efficient schedule that satisfies this requirement unless it uses FSTNs. Figure 32-65 illustrates the general layout of the periodic schedule.

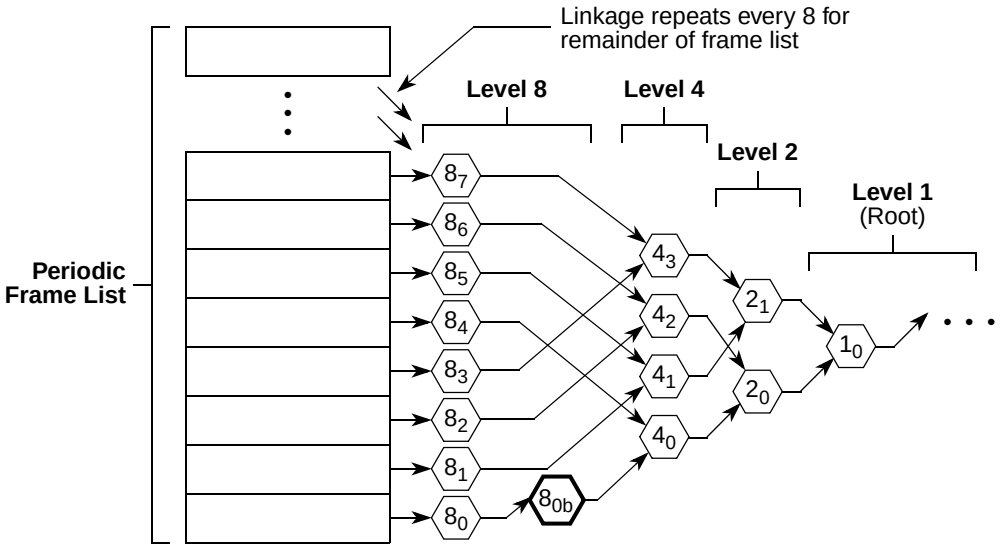


Figure 32-65. General Structure of EHCI Periodic Schedule Utilizing Interrupt Spreading

The periodic frame list is effectively the leaf level a binary tree, which is always traversed leaf to root. Each level in the tree corresponds to a 2^N poll rate. Software can efficiently manage periodic bandwidth on the USB by spreading interrupt queue heads that have the same poll rate requirement across all the available paths from the frame list. For example, system software can schedule eight poll rate eight queue heads and account for them once in the high-speed bus bandwidth allocation.

When an endpoint is allocated an execution footprint that spans a frame boundary, the queue head for the endpoint must be reachable from consecutive locations in the frame list. An example would be if 8_{0b} were such an endpoint. Without additional support on the interface, to get 8_{0b} reachable at the correct time, software would have to link 8_1 to 8_{0b}. It would then have to move 4_1 and everything linked after into the same path as 4_0. This upsets the integrity of the binary tree and disallows the use of the spreading technique.

FSTN data structures are used to preserve the integrity of the binary-tree structure and enable the use of the spreading technique. Section 32.5.7, “Periodic Frame Span Traversal Node (FSTN),” defines the hardware and software operational model requirements for using FSTNs.

The following queue head fields are initialized by system software to instruct the host controller when to execute portions of the split-transaction protocol.

- SplitXState. This is a single bit residing in the status field of a queue head (Table 32-59). This bit is used to track the current state of the split transaction.

- Frame S-mask. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an H-Frame) that the host controller should execute a start-split transaction. This is always qualified by the value of the SplitXState bit in the status field of the queue head. For example, in case one of Figure 32-64, the S-mask would have a value of 0b0000_0001, indicating that if the queue head is traversed by the host controller, the SplitXState indicates Do_Start, and the current micro-frame as indicated by USB_FRINDEX[2:0] is 0 executes a start-split transaction.
- Frame C-mask. This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an H-Frame) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the SplitXState bit in the status field of the queue head. For example, in case one of Figure 32-64, the C-mask would have a value of 0b0001_1100, indicating that if the queue head is traversed by the host controller, and the SplitXState indicates Do_Complete, the current micro-frame as indicated by USB_FRINDEX[2:0] is 2, 3, or 4 executes a complete-split transaction. It is software's responsibility to ensure that the translation between H-Frames and B-Frames is correctly performed when setting bits in S-mask and C-mask.

32.6.11.2.2 Host Controller Operational Model for FSTNs

The FSTN data structure manages low/full-speed interrupt queue heads that need to be reached from consecutive frame list locations (boundary cases 2a through 2c). An FSTN is essentially a back pointer, similar in intent to the back pointer field in the siTD data structure.

This feature provides software a simple primitive to save a schedule position, redirect the host controller to traverse the necessary queue heads in the previous frame, and then restore the original schedule position and complete normal traversal.

There are four components to the use of FSTNs:

- FSTN data structure, defined in Section 32.5.7, “Periodic Frame Span Traversal Node (FSTN).”
- A save place indicator; this is always an FSTN with its back path link pointer [T] bit cleared.
- A restore indicator; this is always an FSTN with its back path link pointer [T] bit set.
- Host controller FSTN traversal rules.

Host Controller Operational Model for FSTNs

When the host controller encounters an FSTN during micro-frames two through seven, it follows the node's normal path link pointer to access the next schedule data structure. The FSTN’s normal path link pointer [T] bit may set, which the host controller must interpret as the end of periodic list mark.

When the host controller encounters a save-place FSTN in micro-frames 0 or 1, it saves the value of the normal path link pointer and sets an internal flag indicating it is executing in recovery path mode. Recovery path mode modifies the host controller's rules for how it traverses the schedule and limits which data structures are considered for execution of bus transactions. The host controller continues executing in recovery path mode until it encounters a restore FSTN or it determines it has reached the end of the micro-frame.

The rules for schedule traversal and limited execution while in recovery path mode are:

- Always follow the normal path link pointer when it encounters an FSTN that is a save-place indicator. The host controller must not recursively follow save-place FSTNs. Therefore, while executing in recovery path mode, it must never follow an FSTN's back path link pointer.
- Do not process an siTD or iTD data structure; simply follow its next link pointer.
- Do not process a QH (Queue Head) whose EPS field indicates a high-speed device; follow its horizontal link pointer.
- When a QH's EPS field indicates a full/low-speed device, the host controller only considers it for execution if its SplitXState is DoComplete (this applies whether the PID code indicates an IN or an OUT). Refer to the *EHCI Specification* for a complete list of additional conditions that must be met in general for the host controller to issue a bus transaction. The host controller must not execute a start-split transaction while executing in recovery path mode. Refer to the *EHCI Specification* for special handling when in recovery path mode.
- Stop traversing the recovery path when it encounters an FSTN that is a restore indicator. The host controller unconditionally uses the saved value of the save-place FSTN's normal path link pointer when returning to the normal path traversal. The host controller must clear the context of executing a recovery path when it restores schedule traversal to the save-place FSTN's normal path link pointer.

  If the host controller determines there is not enough time left in the micro-frame to complete processing of the periodic schedule, it abandons traversal of the recovery path and clears the context of executing a recovery path. The result is the host controller starts traversal at the frame list at the start of the next consecutive micro-frame.

An example traversal of a periodic schedule that includes FSTNs is illustrated in Figure 32-66.

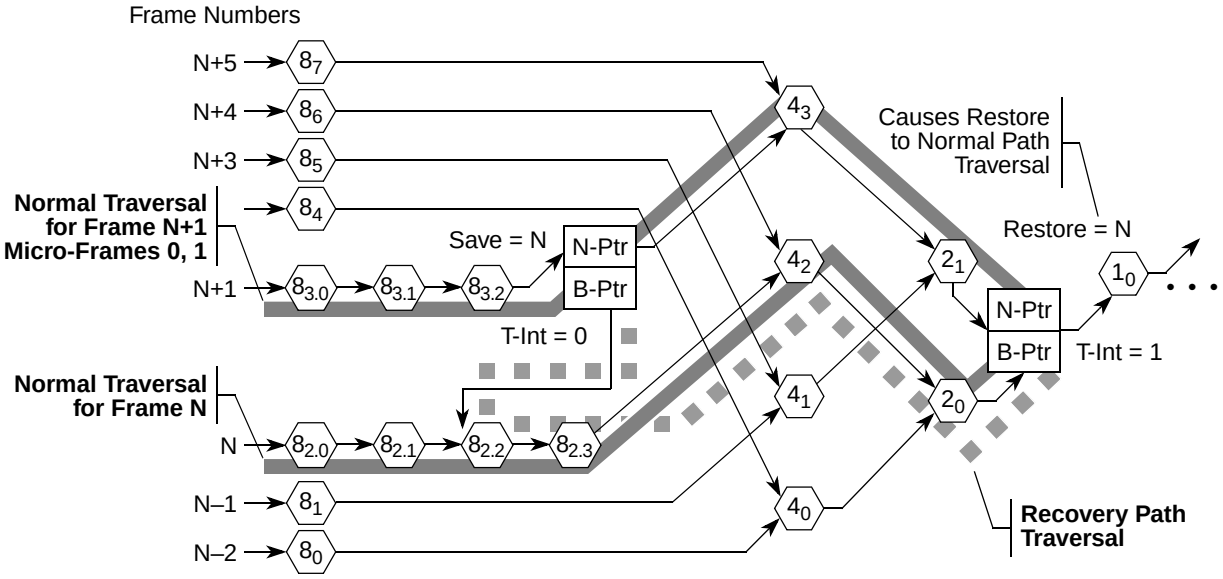


Figure 32-66. Example Host Controller Traversal of Recovery Path via FSTNs

In frame N (micro-frames 0-7), for this example, the host controller traverses all of the schedule data structures utilizing the normal path link pointers in any FSTNs it encounters. This is because the host controller has not yet encountered a save-place FSTN so it is not executing in recovery path mode. When

it encounters the restore FSTN, (Restore-N), during micro-frames 0 and 1, it uses Restore-N. Normal path link pointer to traverse to the next data structure (normal schedule traversal). This is because the host controller must use a restore FSTN's normal path link pointer when not executing in a recovery-path mode. The nodes traversed during frame N include: {$8_{2.0}$, $8_{2.1}$, $8_{2.2}$, $8_{2.3}$, 4_2, 2_0, Restore-N, 1_0...}.

In frame N+1 (micro-frames 0 and 1), when the host controller encounters save-path FSTN (Save-N), it observes that Save-N.Back Path Link Pointer.T-bit is zero (definition of a Save-Path indicator). The host controller saves the value of Save-N. Normal path link pointer and follows Save-N.Back Path Link Pointer. At the same time, it sets an internal flag indicating that it is now in recovery path mode (the recovery path is annotated in Figure 32-66 with a large dashed line). The host controller continues traversing data structures on the recovery path and executing only those bus transactions as noted above, on the recovery path until it reaches restore FSTN (Restore-N). Restore-N.Back Path Link Pointer.T-bit is set (definition of a Restore indicator), so the host controller exits recovery path mode by clearing the internal recovery path mode flag and commences (restores) schedule traversal using the saved value of the save-place FSTN's normal path link pointer (for example, Save-N.Normal Path Link Pointer). The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, $8_{2.2}$, $8_{2.3}$, 4_2, 2_0, Restore-N, 4_3, 2_1, Restore-N, 10...}.

In frame N+1 (micro-frames 2-7), when the host controller encounters save-path FSTN Save-N, it unconditionally follows Save-N.Normal Path Link Pointer. The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, 4_3, 2_1, Restore-N, 1_0...}.

Software Operational Model for FSTNs

Software must create a consistent, coherent schedule for the host controller to traverse. When using FSTNs, system software must adhere to the following rules:

- Each save-place indicator requires a matching restore indicator.

  The save-place indicator is an FSTN with a valid back path link pointer and T-bit equal to zero. The Back path link pointer [Typ] field must be set to indicate the referenced data structure is a queue head. The restore indicator is an FSTN with its back path link pointer [T] bit set.

  A restore FSTN may be matched to one or more save-place FSTNs. For example, if the schedule includes a poll-rate 1 level, system software only needs to place a restore FSTN at the beginning of this list to match all possible save-place FSTNs.
- If the schedule does not have elements linked at a poll-rate level of one, and one or more save-place FSTNs are used, system software must ensure the restore FSTN's normal path link pointer's T-bit is set, as this marks the end of the periodic list.
- When the schedule does have elements linked at a poll rate level of one, a restore FSTN must be the first data structure on the poll rate one list. All traversal paths from the frame list converge on the poll-rate one list. System software must ensure that recovery path mode is exited before the host controller is allowed to traverse the poll rate level one list.
- A save-place FSTN's back path link pointer must reference a queue head data structure. The referenced queue head must be reachable from the previous frame list location. In other words, if the save-place FSTN is reachable from frame list offset N, then the FSTN's back path link pointer must reference a queue head that is reachable from frame list offset N-1.

Software should make the schedule as efficient as possible. What this means in this context is software should have no more than one save-place FSTN reachable in any single frame. Two (or more, depending on the implementation) could exist as full/low-speed footprints change with bandwidth adjustments. This could occur, for example, when a bandwidth rebalance causes system software to move the save-place FSTN from one poll rate level to another. During the transition, software must preserve the integrity of the previous schedule until the new schedule is in place.

32.6.11.2.3 Tracking Split Transaction Progress for Interrupt Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where data is lost. For interrupt-IN transfers, data is lost when it makes it into the USB 2.0 hub, but the USB 2.0 host system is unable to get it from the USB 2.0 hub and into the system before it expires from the transaction translator pipeline. When a lost data condition is detected, the queue is halted, thus signaling system software to recover from the error. A data-loss condition exists when a start-split is issued, accepted, and successfully executed by the USB 2.0 hub, but the complete-splits get unrecoverable errors on the high-speed link or the complete-splits do not occur at the correct times. One reason complete-splits might not occur at the right time would be due to host-induced system hold-offs that cause the host controller to miss bus transactions because it cannot get timely access to the schedule in system memory.

The same condition can occur for an interrupt-OUT, but the result is not an endpoint halt condition. Instead, it affects only the progress of the transfer. The queue head has the following fields to track the progress of each split transaction. These fields are used to keep incremental state about which (and when) portions have been executed.

- C-prog-mask. This is an 8-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the transaction translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. C-prog-mask is a simple bit-vector the host controller sets as one of the C-prog-mask bits for each complete-split executed. The bit position is determined by the micro-frame number in which the complete-split was executed. The host controller always checks C-prog-mask before executing a complete-split transaction. If the previous complete-splits have not been executed then it means one (or more) have been skipped and data has potentially been lost.
- FrameTag. The host controller uses his field during the complete-split portion of the split transaction to tag the queue head with the frame number (H-Frame number) when the next complete split must be executed.
- S-bytes. This field can be used to store the number of data payload bytes sent during the start-split (if the transaction was an OUT). The S-bytes field must be used to accumulate the data payload bytes received during the complete-splits (for an IN).

32.6.11.2.4 Split Transaction Execution State Machine for Interrupt

In the following section, all references to micro-frame are in the context of a micro-frame within an H-Frame.

As with asynchronous full- and low-speed endpoints, a split-transaction state machine manages the split transaction sequence. Aside from the fields defined in the queue head for scheduling and tracking the split transaction, the host controller calculates one internal mechanism that also manages the split transaction. The internal calculated mechanism is:

- cMicroFrameBit. This is a single-bit encoding of the current micro-frame number. It is an eight-bit value calculated by the host controller at the beginning of every micro-frame. It is calculated from the three least significant bits of the USB_FRINDEX register, which is cMicroFrameBit = 1 shifted-left(USB_FRINDEX[2:0]). The cMicroFrameBit has at most one bit asserted that always corresponds to the current micro-frame number. For example, if the current micro-frame is 0, then cMicroFrameBit equals 0b0000_0001.

  The variable cMicroFrameBit is used to compare against the S-mask and C-mask fields to determine whether the queue head is marked for a start- or complete-split transaction for the current micro-frame.

Figure 32-67 illustrates how a complete interrupt-split transaction is managed. There are two phases to each split transaction. The first is a single start-split transaction that occurs when the SplitXState is at Do_Start and the single bit in cMicroFrameBit has a corresponding bit active in QH[S-mask]. The transaction translator does not acknowledge the receipt of the periodic start-split, so the host controller unconditionally transitions the state to Do_Complete. Due to the available jitter in the transaction translator pipeline, there is more than one complete-split transaction scheduled by software for the Do_Complete state. This translates simply to the fact that there are multiple bits set in the QH[C-mask] field.

The host controller keeps the queue head in the Do_Complete state until the split transaction is complete (see definition below), or an error condition triggers the three-strikes-rule (for example, after the host tries the same transaction three times, and each encounters an error, the host controller stops retrying the bus transaction and halts the endpoint, therefore, requiring system software to detect the condition and perform system-dependent recovery).

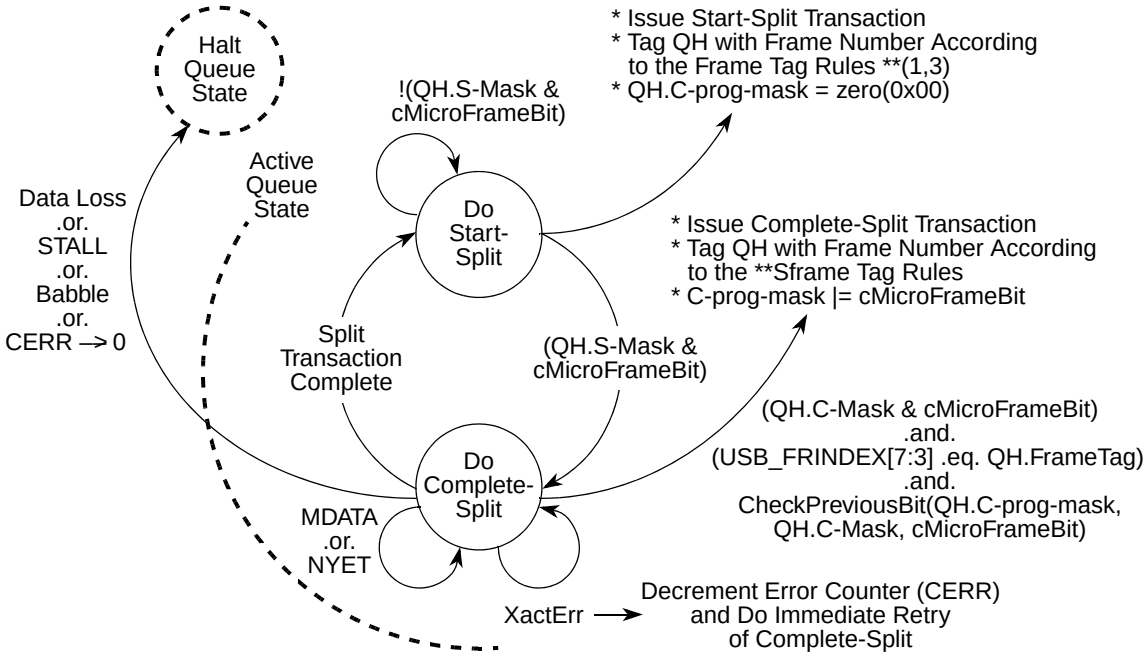


Figure 32-67. Split Transaction State Machine for Interrupt

Periodic Interrupt—Do-Start-Split

This is the state software must initialize a full- or low-speed interrupt queue head StartXState bit. This state is entered from the Do_Complete Split state only after the split transaction is complete. This occurs when one of the following events occur:

- NAK. A NAK response is a propagation of the full- or low-speed endpoint's NAK response.
- ACK. An ACK response is a propagation of the full- or low-speed endpoint's ACK response. Only occurs on an OUT endpoint.
- DATA 0/1. Only occurs for INs. Indicates this is the last of the data from the endpoint for this split transaction.
- ERR. The transaction on the low-/full-speed link below the transaction translator had a failure (for example, timeout, bad CRC, etc.).
- NYET (and Last). The host controller issued the last complete-split and the transaction translator responded with a NYET handshake. This means the start-split was not correctly received by the transaction translator, so it never executed a transaction to the full- or low-speed endpoint, see Section 32.6.11.2, “Split Transaction Interrupt.”

Each time the host controller visits a queue head in this state (within the execute transaction state), bit-wise ANDs QH[S-mask] with cMicroFrameBit to determine whether to execute a start-split. If the result is non-zero, the host controller issues a start-split transaction. If the PID Code field indicates an IN transaction, the host controller must zero-out the QH[S-bytes] field. After the split-transaction has been executed, the host controller sets up state in the queue head to track the progress of the complete-split phase of the split transaction. Specifically, it records the expected frame number into QH[FrameTag] field, sets C-prog-mask to zero (0x00), and exits this state. The host controller must not adjust the value of Cerr as a result of completion of a start-split transaction.

Periodic Interrupt—Do-Complete-Split

This state is entered unconditionally from the Do Start Split state after a start-split transaction is executed on the bus. Each time the host controller visits a queue head in this state (within the execute transaction state), it checks to determine whether a complete-split transaction should be executed now.

There are four tests to determine whether a complete-split transaction should be executed.

- Test A. cMicroFrameBit is bit-wise ANDed with QH[C-mask] field. A non-zero result indicates software scheduled a complete-split for this endpoint, during this micro-frame.
- Test B. QH[FrameTag] is compared with the current contents of USB_FRINDEX[7:3]. An equal indicates a match.
- Test C. The complete-split progress bit vector is checked to determine whether the previous bit is set, indicating the previous complete-split was appropriately executed. An example algorithm for this test is provided below:

```
Algorithm Boolean CheckPreviousBit(QH.C-prog-mask, QH.C-mask, cMicroFrameBit)
Begin
-- Return values:
-- TRUE - no error
-- FALSE - error
--
Boolean rvalue = TRUE;
previousBit = cMicroframeBit logical-rotate-right(1)
-- Bit-wise anding previousBit with C-mask indicates
-- whether there was an intent
-- to send a complete split in the previous micro-frame. So,
-- if the
-- 'previous bit' is set in C-mask, check C-prog-mask to
-- make sure it
-- happened.
If (previousBit bitAND QH.C-mask)then
                                If not(previousBit bitAND QH.C-prog-mask) then
         rvalue = FALSE;
    End if
End If
-- If the C-prog-mask already has a one in this bit position,
-- then an aliasing
-- error has occurred. It probably get caught by the
-- FrameTag Test, but
-- at any rate it is an error condition that as detectable here
-- should not allow
-- a transaction to be executed.
If (cMicroFrameBit bitAND QH.C-prog-mask) then
rvalue = FALSE;
End if
return (rvalue)
End Algorithm
```

- Test D. Check if a start-split should be executed in this micro-frame. This is the same test performed in the Do Start Split state. When it evaluates to TRUE and the controller is NOT processing in the context of a recovery path mode, it means a start-split should occur in this micro-frame. Test D and Test A evaluating to TRUE at the same time is a system software error. Behavior is undefined.

If test A is non-zero, Test B indicates a match event, Test C indicates the previous complete-split was appropriately executed and Test D evaluates to a NOT TRUE condition, the host controller executes a complete-split transaction. When the host controller commits to executing the complete-split transaction, it updates QH[C-prog-mask] by bit-ORing with cMicroFrameBit. On completion of the complete-split transaction, the host controller records the result of the transaction in the queue head and sets QH[FrameTag] to the expected H-Frame number. The effect to the state of the queue head and thus the state of the transfer depends on the response by the transaction translator to the complete-split transaction. The following responses have the effects (any responses that result in decrementing of the Cerr result in the queue head being halted by the host controller if the result of the decrement is zero):

- NYET (and Last). On each NYET response, the host controller checks to determine whether this is the last complete-split for this split transaction. Last is defined in this context as the condition where all of the scheduled complete-splits have been executed. If it is the last complete-split (with a NYET response), the transfer state of the queue head is not advanced (never received any data) and this state exited. The transaction translator must have responded to all the complete-splits with NYETs, meaning the start-split issued by the host controller was not received. The start-split should be retried at the next poll period.
- The test for whether this is the last complete split can be performed by XOR QH[C-mask] with QH[C-prog-mask]. If the result is all zeros, all complete-splits have been executed. When this condition occurs, the XactErr status bit is set and the Cerr field is decremented.
- NYET (and not Last). See above description for testing for last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for C-prog-mask and FrameTag) and stay in this state. The host controller must not adjust Cerr on this response.
- Transaction Error (XactErr). Timeout, data CRC failure, etc. The Cerr field is decremented and the XactErr bit in the Status field is set. The complete split transaction is immediately retried (if Cerr is non-zero).If there is not enough time in the micro-frame to complete the retry and the endpoint is an IN or Cerr is decremented to a zero from a one, the queue is halted. If there is not enough time in the micro-frame to complete the retry and the endpoint is an OUT and Cerr is not zero, this state is exited (return to Do Start Split). This results in a retry of the entire OUT split transaction at the next poll period. Refer to Chapter 11 Hubs (specifically the section on full- and low-speed interrupts) in the *USB Specification Revision 2.0* for detailed requirements on why these errors must be immediately retried.
- ACK. This can only occur if the target endpoint is an OUT. The target endpoint ACKed the data and this response is a propagation of the endpoint ACK to the host controller. The host controller must advance the state of the transfer. The current offset field is incremented by maximum packet length or bytes to transfer, whichever is less. The bytes to transfer field is decremented by the same amount, and the data toggle bit (dt) is toggled. The host controller then exits this state for this queue head. The host controller must reload Cerr with maximum value on this response. Advancing the transfer state may cause other process events such as retirement of the qTD and advancement of the queue.
- MDATA. This response occurs only for an IN endpoint. The transaction translator responded with zero or more bytes of data and an MDATA PID. The incremental number of bytes received is accumulated in QH[S-bytes]. The host controller must not adjust Cerr on this response.

- DATA0/1. This response may only occur for an IN endpoint. The number of bytes received is added to the accumulated byte count in QH[S-bytes]. The state of the transfer is advanced by the result and the host controller exits this state for this queue head.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue.
- If the data sequence PID does not match the expected, the entirety of the data received in this split transaction is ignored, the transfer state is not advanced, and this state is exited.
- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced, and this state is exited. The host controller must reload Cerr with maximum value on this response.
- ERR. There was an error during the full- or low-speed transaction. The ERR status bit is set, Cerr is decremented, the state of the transfer is not advanced, and this state is exited.
- STALL. The queue is halted (an exit condition of the Execute Transaction state). The status field bits:

  Active bit is cleared and the Halted bit is set and the qTD is retired. Responses not enumerated in the list or received out of sequence are illegal and may result in undefined host controller behavior. The other possible combinations of tests A, B, C, and D may indicate data or response was lost. Table 32-76 lists the possible combinations and the appropriate action.

Table 32-76. Interrupt IN/OUT Do Complete Split State Execution Criteria

Condition	Action	Description
not(A) not(D)	Ignore QHD	Neither a start nor complete-split is scheduled for the current micro-frame. Host controller should continue walking the schedule.
A not(C)	If PIDCode = IN Halt QHDIf PIDCode = OUT Retry start-split	Progress bit check failed. These means a complete-split has been missed. There is the possibility of lost data. If PID Code is an IN, the queue head must be halted. If PID code is an OUT, the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not affect Cerr. In either case, set the missed micro-frame bit in the status field to a one.
A not(B) C	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	QH.FrameTag test failed. This means that exactly one or more H-Frames have been skipped. This means complete-splits and have missed. There is the possibility of lost data. If PID Code is an IN, then the Queue head must be halted.If PID Code is an OUT, the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not affect Cerr. In either case, set the missed micro-frame bit in the status field to a one.
A B C not(D)	Execute complete-split	This is the non-error case where the host controller executes a complete-split transaction.

Table 32-76. Interrupt IN/OUT Do Complete Split State Execution Criteria (continued)

Condition	Action	Description
D	If PIDCode = IN Halt QHDIf PIDCode = OUT Retry start-split	This is a degenerate case where the start-split was issued, but all of the complete-splits were skipped and all possible intervening opportunities to detect the missed data failed to fire. If PID Code is an IN, the queue head must be halted. If PID code is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not affect Cerr. In either case, set the missed micro-frame bit in the status field to a one. When executing in the context of a recovery path mode, the host controller can process the queue head and take the actions indicated above or it may wait until the queue head is visited in the normal processing mode. Regardless, the host controller must not execute a start-split in the context of an executing in a recovery path mode.

Managing the QH[FrameTag] Field

The QH[FrameTag] field in a queue head is completely managed by the host controller. The rules for setting QH[FrameTag] are simple:

- Rule 1: If transitioning from Do Start Split to Do Complete Split and the current value of USB_FRINDEX[2:0] is 6, QH[FrameTag] is set to USB_FRINDEX[7:3] + 1. This accommodates split transactions whose start-split and complete-splits are in different H-Frames (case 2a, see Figure 32-64).
- Rule 2: If the current value of USB_FRINDEX[2:0] is 7, QH[FrameTag] is set to USB_FRINDEX[7:3] + 1. This accommodates staying in Do Complete Split for cases 2a, 2b, and 2c in Figure 32-64.
- Rule 3: If transitioning from Do_Start Split to Do Complete Split and the current value of USB_FRINDEX[2:0] is not 6, or currently in Do Complete Split and the current value of (USB_FRINDEX[2:0]) is not 7, FrameTag is set to USB_FRINDEX[7:3]. This accommodates all other cases in Figure 32-64.

32.6.11.2.5 Rebalancing the Periodic Schedule

System software must occasionally adjust a periodic queue head's S-mask and C-mask fields during operation. This need occurs when adjustments to the periodic schedule create a new bandwidth budget and one or more queue head's are assigned new execution footprints (new S-mask and C-mask values).

It is imperative that system software not update these masks to new values in the midst of a split transaction. To avoid any race conditions with the update, the host controller provides a simple assist to system software. System software sets the inactivate-on-next-transaction (I) bit to signal the host controller it intends to update the S-mask and C-mask on this queue head. System software then waits for the host controller to observe the I-bit is set and transitions the active bit to a zero. The rules for how and when the host controller clears the active bit are:

- If the active bit is cleared, no action is taken. The host controller does not attempt to advance the queue when the I-bit is set.
- If the active bit is set and the SplitXState is DoStart (regardless of the value of S-mask), the host controller simply clears the active bit. The host controller is not required to write the transfer state

back to the current qTD. If the S-mask indicates that a start-split is scheduled for the current micro-frame, the host controller must not issue the start-split bus transaction; it must clear the active bit.

System software must save transfer state before setting the I-bit. This is required so it can correctly determine what transfer progress (if any) occurred after the I-bit was set, and the host controller executed it's final bus-transaction and cleared the active bit.

After system software has updated the S-mask and C-mask, it must reactivate the queue head. Because the active bit and the I-bit cannot be updated with the same write, system software needs to use the following algorithm to coherently re-activate a queue head that has been stopped using the I-bit.

1. Set the halted bit
2. Clear the I-bit
3. Set the active bit and clear the halted bit in the same write.

Setting the halted bit inhibits the host controller from attempting to advance the queue between the time the I-bit is cleared and the active bit is set.

32.6.11.3 Split Transaction Isochronous

Full-speed isochronous transfers are managed using the split-transaction protocol through a USB 2.0 transaction translator in a USB 2.0 hub. The host controller uses siTD data structure to support the special requirements of isochronous split-transactions. This data structure uses the scheduling model of isochronous TDs (see Section 32.6.6, “Managing Isochronous Transfers Using iTDs,” for the operational model of iTDs) with the contiguous data feature provided by queue heads. This simple arrangement allows a single isochronous scheduling model and adds the additional feature that all data received from the endpoint (per split transaction) must land into a contiguous buffer.

32.6.11.3.1 Split Transaction Scheduling Mechanisms for Isochronous Transactions

Full-speed isochronous transactions are managed through a transaction translator's periodic pipeline. As with full- and low-speed interrupt, system software manages each transaction translator's periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each full-speed isochronous endpoint occur. The requirements described in Section 32.6.11.2.1, “Split Transaction Scheduling Mechanisms for Interrupt,” apply. Figure 32-68 illustrates the general scheduling boundary conditions supported by the EHCI periodic schedule. The S_n and C_n labels indicate micro-frames where software can schedule start- and complete-splits (respectively). The H-Frame boundaries are marked with a large, solid bold vertical line. The B-Frame boundaries are marked with a large, bold, and dashed line. The bottom of the figure illustrates the relationship of an siTD to the H-Frame.

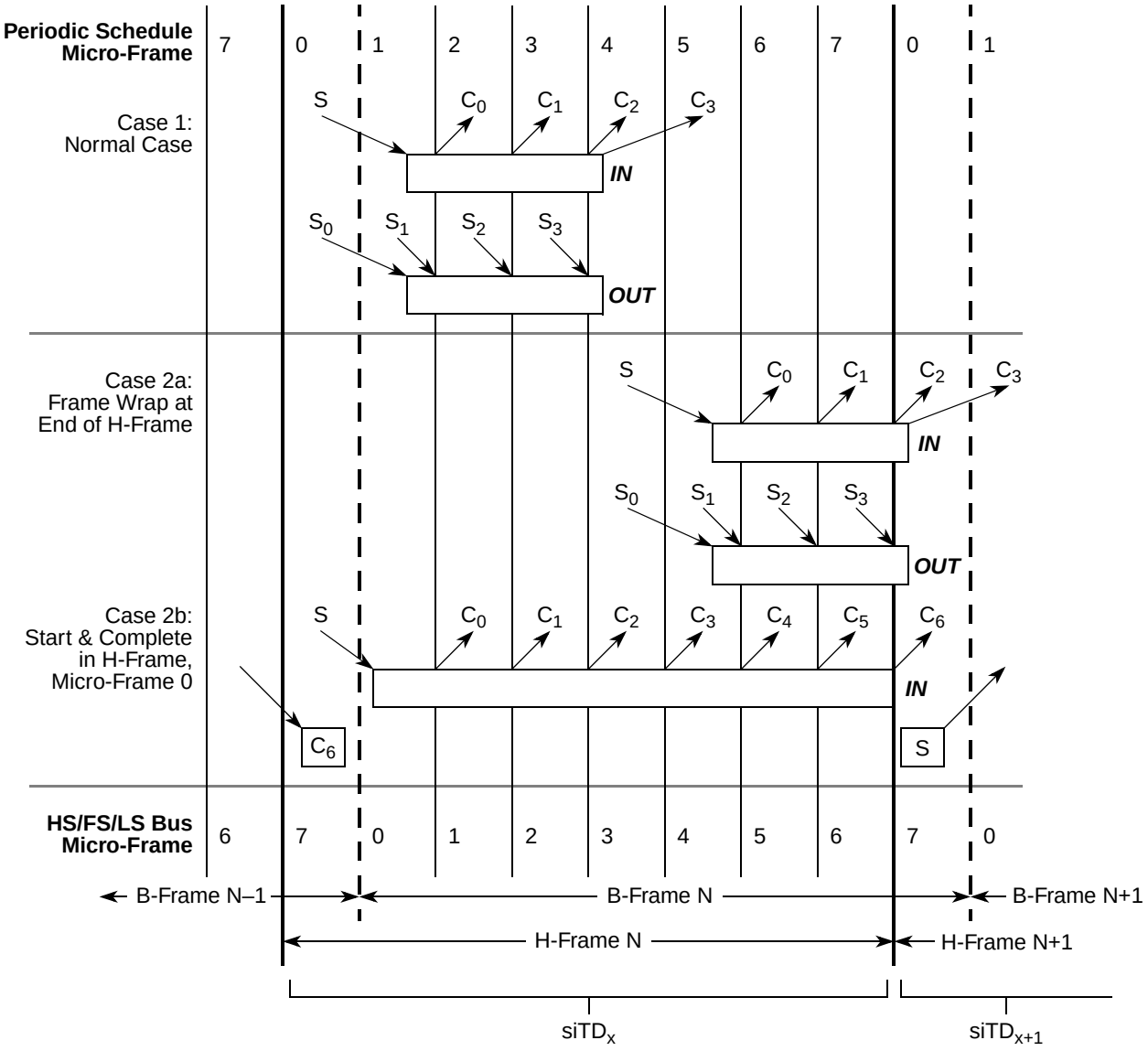


Figure 32-68. Split Transaction and Isochronous Scheduling Boundary Conditions

When the endpoint is an isochronous OUT, there are only start-splits and no complete-splits. When the endpoint is an isochronous IN, there is at most one start-split and one to N complete-splits. The scheduling boundary cases are:

- Case 1: The entire split transaction is completely bounded by an H-Frame. For example, the start-splits and complete-splits are all scheduled to occur in the same H-Frame.
- Case 2a: This boundary case is where one or more (at most two) complete-splits of a split transaction IN are scheduled across an H-Frame boundary. This can only occur when the split transaction has the possibility of moving data in B-Frame, micro-frames 6 or 7 (H-Frame micro-frame 7 or 0). When an H-Frame boundary wrap condition occurs, the scheduling of the split transaction spans more than one location in the periodic list (for example, it takes two siTDs in adjacent periodic frame list locations to fully describe the scheduling for the split transaction).

Although the scheduling of the split transaction may take two data structures, all of the complete-splits for each full-speed IN isochronous transaction must use only one data pointer. For this reason, siTDs contain a back pointer.

Software must never schedule full-speed isochronous OUTs across an H-Frame boundary.

- Case 2b: This case can only occur for a large isochronous IN. It is the only allowed scenario where a start-split and complete-split for the same endpoint can occur in the same micro-frame. Software must enforce this rule by scheduling the large transaction first. Large is defined to be anything larger than 579 byte maximum packet size.

A subset of the same mechanisms employed by full- and low-speed interrupt queue heads are employed in siTDs to schedule and track the portions of isochronous split transactions. The following fields are initialized by system software to instruct the host controller when to execute portions of the split transaction protocol:

- SplitXState. This is a single bit residing in the status field of an siTD (see Table 32-53). This bit is used to track the current state of the split transaction. The rules for managing this bit are described in Section 32.6.11.3.3, “Split Transaction Execution State Machine for Isochronous Transactions.”
- Frame S-mask. This is a bit-field wherein system software sets a bit corresponding to the micro-frame (within an H-Frame) that the host controller should execute a start-split transaction. This is always qualified by the value of the SplitXState bit. For example, referring to the IN example in case one of Figure 32-68, the S-mask would have a value of 0b0000_0001 indicating that if the siTD is traversed by the host controller, the SplitXState indicates Do Start Split, and the current micro-frame as indicated by USB_FRINDEX[2:0] is 0, then execute a start-split transaction.
- Frame C-mask. This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an H-Frame) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the SplitXState bit. For example, referring to the IN example in case one of Figure 32-68, the C-mask would have a value of 0b 0011_1100 indicating that if the siTD is traversed by the host controller, the SplitXState indicates Do Complete Split, and the current micro-frame as indicated by USB_FRINDEX[2:0] is 2, 3, 4, or 5, then execute a complete-split transaction.
- Back Pointer. This field in an siTD is used to complete an IN split-transaction using the previous H-Frame's siTD. This is only used when the scheduling of the complete-splits span an H-Frame boundary.

There exists a one-to-one relationship between a high-speed isochronous split transaction (including all start- and complete-splits) and one full-speed isochronous transaction. An siTD contains (amongst other things) buffer state and split transaction scheduling information. An siTD's buffer state always maps to one full-speed isochronous data payload. This means that for any full-speed transaction payload, a single siTD's data buffer must be used. This rule applies to both IN an OUTs. An siTD's scheduling information usually also maps to one high-speed isochronous split transaction. The exception to this rule is the H-Frame boundary wrap cases mentioned above.

The siTD data structure describes at most, one frame's worth of high-speed transactions and that description is strictly bounded within a frame boundary. At the top of Figure 32-69 are examples of the full-speed transaction footprints for the boundary scheduling cases described above. In the middle are

time-frame references for both the B-Frames (HS/FS/LS Bus) and the H-Frames. At the bottom is illustrated the relationship between the scope of an siTD description and the time references. Each H-Frame corresponds to a single location in the periodic frame list. The implication is that each siTD is reachable from a single periodic frame list location at a time.

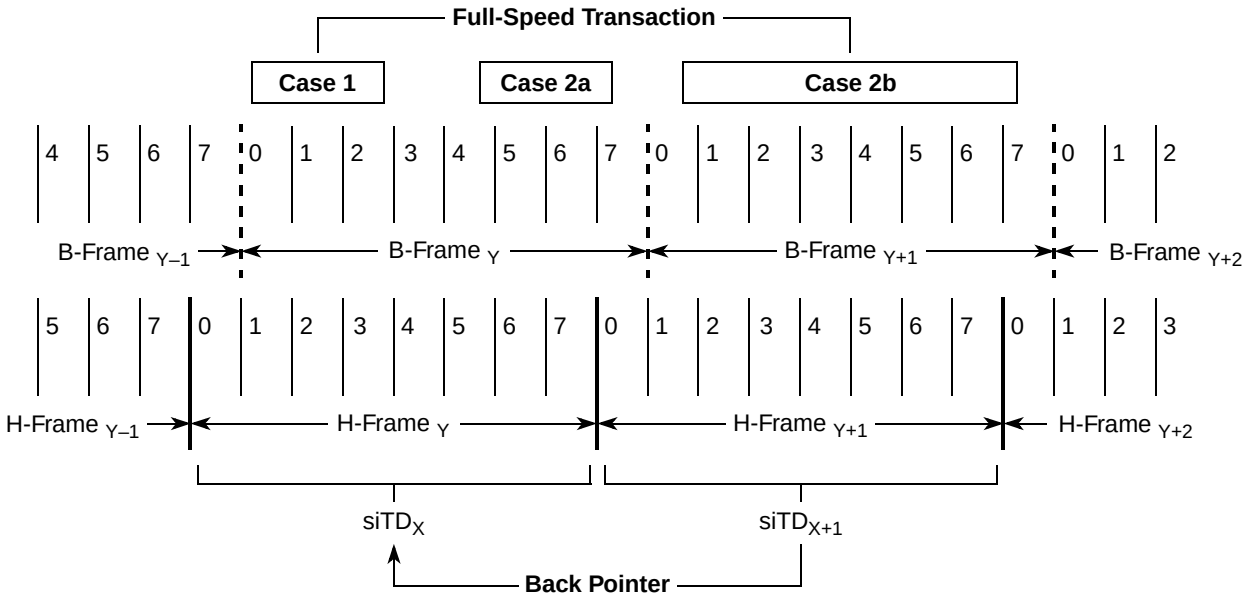


Figure 32-69. siTD Scheduling Boundary Examples

Each case is described below:

- Case 1: One siTD is sufficient to describe and complete the isochronous split transaction because the whole isochronous split transaction is tightly contained within a single H-Frame.
- Case 2a, 2b: Although both INs and OUTs can have these footprints, OUTs always take only one siTD to schedule. However, INs (for these boundary cases) require two siTDs to complete the scheduling of the isochronous split transaction. The siTDX is used to always issue the start-split and the first N complete-splits. The full-speed transaction (for these cases) can deliver data on the full-speed bus segment during micro-frame 7 of H-Frame$_{Y+1}$, or micro-frame 0 of H-Frame$_{Y+2}$. The complete splits are scheduled using siTD$_{X+2}$ (not shown). The complete-splits to extract this data must use the buffer pointer from siTD$_{X+1}$. The only way for the host controller to reach siTD$_{X+1}$ from H-Frame$_{Y+2}$ is to use siTD$_{X+2}$'s back pointer.

Software must apply the following rules when calculating the schedule and linking the schedule data structures into the periodic schedule:

- Software must ensure that an isochronous split-transaction is started so it completes before the end of the B-Frame.
- Software must ensure that for a single full-speed isochronous endpoint, there is never a start-split and complete-split in H-Frame, micro-frame 1. This is mandated as a rule so that case 2a and case 2b can be discriminated. According to the core USB specification, the long isochronous transaction illustrated in Case 2b could be scheduled so the start-split was in micro-frame 1 of H-Frame N and the last complete-split would need to occur in micro-frame 1 of H-Frame N+1. However, it is

impossible to discriminate between cases 2a and 2b, which has significant impact on the complexity of the host controller.

32.6.11.3.2 Tracking Split Transaction Progress for Isochronous Transfers

Isochronous endpoints do not employ the concept of a halt on error; however, the host controller does identify and report per-packet errors observed in the data stream. This includes schedule traversal problems (skipped micro-frames), timeouts, and corrupted data received.

In similar kind to interrupt split-transactions, the portions of the split transaction protocol must execute in the micro-frames they are scheduled. The queue head data structure that manages full- and low-speed interrupt has several mechanisms for tracking when portions of a transaction have occurred. Isochronous transfers use siTDs for their transfers and the data structures are only reachable using the schedule in the exact micro-frame in which they are required (so all the mechanism employed for tracking in queue heads is not required for siTDs). Software has the option of reusing siTD several times in the complete periodic schedule. However, it must ensure that the results of split transaction N are consumed and the siTD re-initialized (activated) before the host controller gets back to the siTD (in a future micro-frame).

Split-transaction isochronous OUTs utilize a low-level protocol to indicate which portions of the split transaction data have arrived. Control over the low-level protocol is exposed in an siTD using the fields transaction position (TP) and transaction count (T-count). If the entire data payload for the OUT split transaction is larger than 188 bytes, there is more than one start-split transaction, each of which requires proper annotation. If host hold-offs occur, the sequence of annotations received from the host is not complete, which is detected and managed by the transaction translator. See Section 32.6.11.3.1, “Split Transaction Scheduling Mechanisms for Isochronous Transactions,” for a description on how these fields are used during a sequence of start-split transactions.

The fields siTD[T-Count] and siTD[TP] are used by the host controller to drive and sequence the transaction position annotations. It is the responsibility of system software to properly initialize these fields in each siTD. After the budget for a split-transaction isochronous endpoint is established, S-mask, T-Count, and TP initialization values for all the siTD associated with the endpoint are constant. They remain constant until the budget for the endpoint is recalculated by software and the periodic schedule adjusted.

For IN-endpoints, the transaction translator simply annotates the response data packets with enough information to allow the host controller to identify the last data. As with split-transaction interrupt, it is the host controller's responsibility to detect when it has missed an opportunity to execute a complete-split. The following field in the siTD is used to track and detect errors in the execution of a split transaction for an IN isochronous endpoint.

- C-prog-mask. This is an 8-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the transaction translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. C-prog-mask is a simple bit-vector that the host controller sets a bit for each complete-split executed. The bit position is determined by the micro-frame (USB_FRINDEX[2:0]) number in which the complete-split was executed. The host controller always checks C-prog-mask before executing a complete-split

transaction. If the previous complete-splits have not been executed, it means one (or more) have been skipped and data has potentially been lost. System software is required to initialize this field to zero before setting an siTD's active bit to a one.

If a transaction translator returns with the final data before all of the complete-splits have been executed, the state of the transfer is advanced so the remaining complete-splits are not executed. An IN siTD is retired based solely on the responses from the transaction translator to the complete-split transactions. This means, for example, it is possible for a transaction translator to respond to a complete-split with an MDATA PID. The number of bytes in the MDATA's data payload could cause the siTD [total bytes to transfer] field to decrement to zero. This response can occur, before all of the scheduled complete-splits have been executed. In another interface such as data structures (for example, high-speed data streams through queue heads), the transition of total bytes to transfer to zero signals the end of the transfer and results in clearing the active bit. However, in this case, the result has not been delivered by the transaction translator and the host must continue with the next complete-split transaction to extract the residual transaction state. This scenario occurs because of the pipeline rules for a transaction translator. In summary, the periodic pipeline rules require that on a micro-frame boundary, the transaction translator holds the final two bytes received (if it has not seen an end of packet [EOP]) in the full-speed bus pipe stage and gives the remaining bytes to the high-speed pipeline stage. At the micro-frame boundary, the transaction translator could have received the entire packet (including both CRC bytes), but not received the packet EOP. In the next micro-frame, the transaction translator responds with an MDATA and sends all of the data bytes (with the two CRC bytes held in the full-speed pipeline stage). This could cause the siTD to decrement it's total bytes to transfer field to zero, indicating it has received all expected data. The host must continue to execute one more (scheduled) complete-split transactions to extract the results of the full-speed transaction from the transaction translator (for example, the transaction translator may have detected a CRC failure, and this result must be forwarded to the host).

If the host experiences hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous OUT, the protocol to the transaction translator is not consistent and the transaction translator detects and reacts to the problem. Likewise, for host hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous IN, the C-prog-mask is used by the host controller to detect errors. However, if the host experiences a hold-off that causes it to skip all of an siTD, or an siTD expires during a host hold off (for example, a hold-off occurs and the siTD is no longer reachable by the host controller for it to report the hold-off event), system software must detect the siTDs have not been processed by the host controller (for example, state not advanced) and report the appropriate error to the client driver.

32.6.11.3.3 Split Transaction Execution State Machine for Isochronous Transactions

In this section, all references to micro-frame are in the context of a micro-frame within an H-Frame.

If the active bit in the status byte is a zero, the host controller ignores the siTD and continues traversing the periodic schedule. Otherwise, the host controller processes the siTD as specified below. A split transaction state machine manages the split-transaction protocol sequence. The host controller uses the fields defined in Section 32.6.11.3.2, "Tracking Split Transaction Progress for Isochronous Transfers," plus the variable cMicroFrameBit defined in Section 32.6.11.2.4, "Split Transaction Execution State Machine for Interrupt," to track the progress of an isochronous split transaction. Figure 32-70 illustrates the state machine for managing an siTD through an isochronous split transaction. Bold, dotted circles

denote the state of the Active bit in the Status field of an siTD. The Bold, dotted arcs denote the transitions between these states. Solid circles denote the states of the split transaction state machine and the solid arcs denote the transitions between these states. Dotted arcs and boxes reference actions that take place either as a result of a transition or from being in a state.

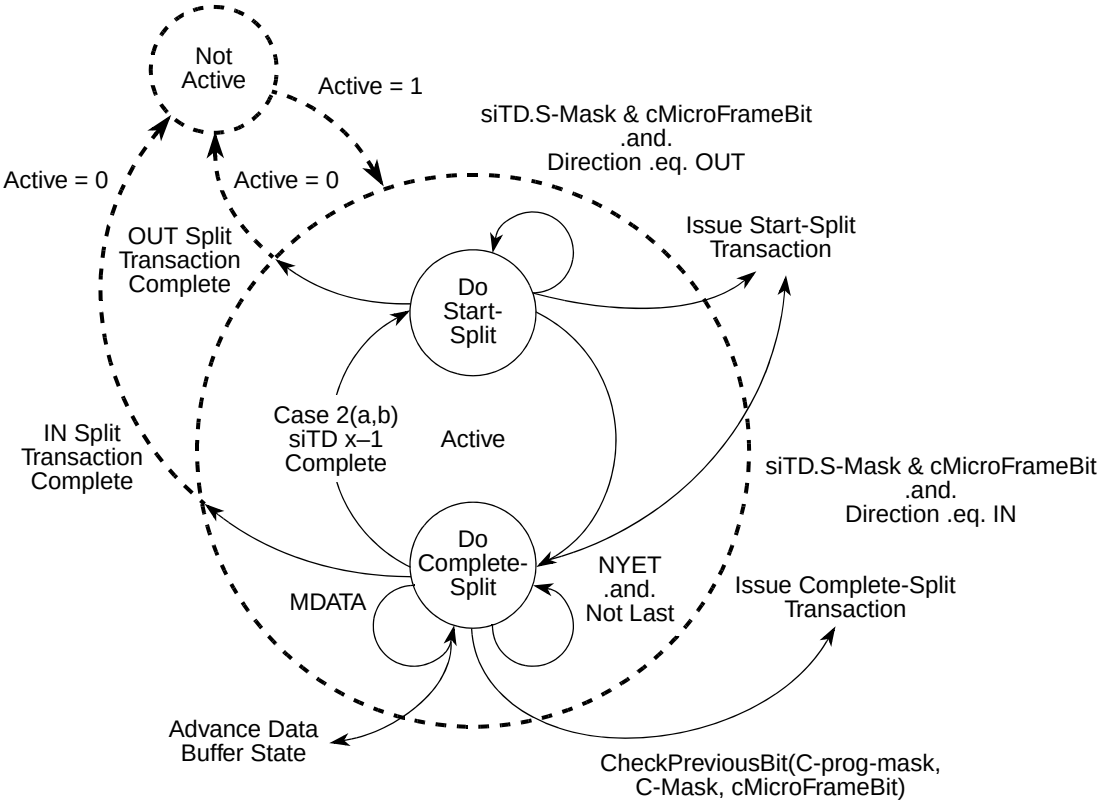


Figure 32-70. Split Transaction State Machine for Isochronous

Periodic Isochronous—Do-Start-Split

Isochronous split transaction OUTs use only this state. An siTD for a split-transaction isochronous IN is initialized to this state, the siTD transitions to this state from Do Complete Split when a case 2a (IN), or 2b scheduling boundary isochronous split-transaction completes.

Each time the host controller reaches an active siTD in this state, it checks the siTD [S-mask] against cMicroFrameBit. If there is a one in the appropriate position, the siTD executes a start-split transaction. By definition, the host controller cannot reach an siTD at the wrong time. If the I/O field indicates an IN, the start-split transaction includes only the extended token plus the full-speed token. Software must initialize the siTD [Total Bytes To Transfer] field to the number of bytes expected. This is usually the maximum packet size for the full-speed endpoint. The host controller exits this state when the start-split transaction is complete.

The remainder of this section is specific to an isochronous OUT endpoint (the I/O field indicates an OUT). When the host controller executes a start-split transaction for an isochronous OUT it includes a data payload in the start-split transaction. The memory buffer address for the data payload is constructed by concatenating siTD [Current Offset] with the page pointer indicated by the page select field (siTD [P]). A

zero in this field selects Page 0 and a one selects Page 1. During the start-split for an OUT, if the data transfer crosses a page boundary during the transaction, the host controller must detect the page cross, update the siTD [P] bit from a zero to a one, and begin using the siTD Page 1 with siTD [Current Offset] as the memory address pointer. The field siTD [TP] is used to annotate each start-split transaction with the indication of which part of the split-transaction data the current payload represents (ALL, BEGIN, MID, END). In all cases, the host controller simply uses the value in siTD [TP] to mark the start-split with the correct transaction position code.

T-Count is always initialized to the number of start-splits for the current frame. TP is always initialized to the first required transaction position identifier. The scheduling boundary case (see Figure 32-69) determines the initial value of TP. The initial cases are summarized in Table 32-77.

Table 32-77. Initial Conditions for OUT siTD's TP and T-count Fields

Case	T-count	TP	Description
1, 2a	=1	ALL	When the OUT data payload is less than (or equal to) 188 bytes, only one start-split is required to move the data. The one start-split must be marked with an ALL.
1, 2a	≠ 1	BEGIN	When the OUT data payload is greater than 188 bytes more than one start-split must be used to move the data. The initial start-split must be marked with a BEGIN.

After each start-split transaction is complete, the host controller updates T-Count and TP appropriately so that the next start-split is correctly annotated. Table 32-78 illustrates all of the TP and T-count transitions, which must be accomplished by the host controller.

Table 32-78. Transaction Position (TP)/Transaction Count (T-Count) Transition Table

TP	T-count next	TP next	Description
ALL	0	N/A	Transition from ALL, to done.
BEGIN	1	END	Transition from BEGIN to END. Occurs when T-count starts at 2.
BEGIN	≠ 1	MID	Transition from BEGIN to MID. Occurs when T-count starts at greater than 2.
MID	≠ 1	MID	TP stays at MID while T-count is not equal to 1 (for example, greater than 1). This case can occur for any of the scheduling boundary cases where the T-count starts greater than 3.
MID	1	END	Transition from MID to END. This case can occur for any of the scheduling boundary cases where the T-count starts greater than 2.

The start-split transactions do not receive a handshake from the transaction translator, so the host controller always advances the transfer state in the siTD after the bus transaction is complete. To advance the transfer state, the following operations take place:

- The siTD [Total Bytes To Transfer] and the siTD [Current Offset] fields are adjusted to reflect the number of bytes transferred.
- The siTD [P] (page select) bit is updated appropriately.
- The siTD [TP] and siTD [T-count] fields are updated appropriately as defined in Table 32-78.

These fields are then written back to the memory based siTD. The S-mask is fixed for the life of the current budget. As mentioned above, TP and T-count are set specifically in each siTD to reflect the data sent from

this siTD. Therefore, regardless of the value of S-mask, the actual number of start-split transactions depends on T-count (or equivalently, total bytes to transfer). The host controller must clear the active bit when it detects all of the schedule data has been sent to the bus. The preferred method is to detect when T-Count decrements to zero as a result of a start-split bus transaction. Equivalently, the host controller can detect when total bytes to transfer decrements to zero. Either implementation must ensure that if the initial condition is total bytes to transfer equals zero and T-count equals one, the host controller issues a single start-split, with a zero-length data payload. Software must ensure that TP, T-count and total bytes to transfer are set to deliver the appropriate number of bus transactions from each siTD. An inconsistent combination yields undefined behavior.

If the host experiences hold-offs that cause the host controller to skip start-split transactions for an OUT transfer, the state of the transfer does not progress appropriately. The transaction translator observes protocol violations in the arrival of the start-splits for the OUT endpoint (the transaction position annotation is incorrect as received by the transaction translator).

Example scenarios are described in Section 32.6.11.3.4, “Split Transaction for Isochronous - Processing Examples.”

The host controller can optionally track the progress of an OUT split transaction by setting appropriate bits in the siTD [C-prog-mask] as it executes each scheduled start-split. The checkPreviousBit algorithm defined in Section , “Periodic Isochronous - Do Complete Split,” can be used prior to executing each start-split to determine if start-splits were skipped. The host controller can use this mechanism to detect missed micro-frames. It can then clear the siTD's active bit and stop execution of this siTD. This saves on both memory and high-speed bus bandwidth.

Periodic Isochronous - Do Complete Split

This state is only used by a split-transaction isochronous IN endpoint. This state is entered unconditionally from the Do Start State after a start-split transaction is executed for an IN endpoint. Each time the host controller visits an siTD in this state, it conducts a number of tests to determine whether it should execute a complete-split transaction. The individual tests are listed below. The sequence they are applied to depends on which micro-frame the host controller is currently executing, which means the tests might not be applied until after the siTD referenced from the back pointer has been fetched.

- Test A. cMicroFrameBit is bit-wise ANDed with the siTD [C-mask] field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame. This test is always applied to a newly fetched siTD in this state.
- Test B. The siTD [C-prog-mask] bit vector is checked to determine whether the previous complete splits have been executed. An example algorithm is given below (this is slightly different than the algorithm used in Periodic Interrupt—Do-Complete-Split). The sequence in which this test is applied depends on the current value of USB_FRINDEX[2:0]. If USB_FRINDEX[2:0] is 0 or 1, it is not applied until the back pointer has been used. Otherwise, it is applied immediately.

```
Algorithm Boolean CheckPreviousBit(siTD.C-prog-mask, siTD.C-mask, cMicroFrameBit)
Begin
    Boolean rvalue = TRUE;
    previousBit = cMicroFrameBit rotate-right(1)
    -- Bit-wise anding previousBit with C-mask indicates whether there
    -- was an intent to send a complete split in the previous micro-
    -- frame. So, if the 'previous bit' is set in C-mask, check
```

```
        -- C-prog-mask to make sure it happened.
        if previousBit bitAND siTD.C-mask then
            if not (previousBit bitAND siTD.C-prog-mask) then
                                                                rvalue = FALSE
            End if
        End if
        Return rvalue
    End Algorithm
```

If Test A is true and USB_FRINDEX[2:0] is zero or one, this is a case 2a or 2b scheduling boundary (see Figure 32-68). See Complete-Split for Scheduling Boundary Cases 2a, 2b for details in managing this condition.

If Test A and Test B evaluate to true, the host controller executes a complete-split transaction using the transfer state of the current siTD. When the host controller commits to executing the complete-split transaction, it updates QH[C-prog-mask] by bit-ORing with cMicroFrameBit. The transfer state is advanced based on the completion status of the complete-split transaction. To advance the transfer state of an IN siTD, the host controller must:

- Decrement the number of bytes received from siTD [Total Bytes To Transfer]
- Adjust siTD [Current Offset] by the number of bytes received
- Adjust the siTD [P] (page select) field if the transfer caused the host controller to use the next page pointer
- Set any appropriate bits in the siTD [Status] field, depending on the results of the transaction.

If the host controller encounters a condition where siTD [total bytes to transfer] is zero and it receives more data, the host controller must not write the additional data to memory. The siTD [Status-Active] bit must be cleared and the siTD [Status-Babble Detected] bit must be set. The fields siTD [Total Bytes To Transfer], siTD [Current Offset], and siTD [P] are not required to be updated as a result of this transaction attempt.

The host controller accepts (assuming good data packet CRC and sufficient room in the buffer as indicated by the value of siTD [Total Bytes To Transfer]) MDATA and DATA0/1 data payloads as large as 192 bytes. The host controller may optionally clear siTD [Status-Active] and set siTD [Status-Babble Detected] when it receives MDATA or DATA0/1 with a data payload larger than 192 bytes. The following responses have the noted effects:

- ERR. The full-speed transaction completed with a time-out or bad CRC and this is a reflection of that error to the host. The host controller sets the ERR bit in the siTD [Status] field and clears the active bit.
- Transaction Error (XactErr). The complete-split transaction encounters a timeout, CRC16 failure, etc. The siTD [Status] field XactErr field is set and the complete-split transaction must be retried immediately. The host controller must use an internal error counter to count the number of retries as a counter field is not provided in the siTD data structure. The host controller does not retry more than two times. If the host controller exhausts the retries or the end of the micro-frame occurs, the active bit is cleared.
- DATAx (0 or 1). This response signals the final data for the split transaction has arrived. The transfer state of the siTD is advanced and the active bit is cleared. If the bytes to transfer field has not decremented to zero (including the reception of the data payload in the DATAx response), less

data than was expected, or allowed for was actually received. This short packet event does not set the USBINT status bit in the USB_USBSTS register to a one. The host controller does not detect this condition.

- NYET (and Last). On each NYET response, the host controller also checks to determine whether this is the last complete-split for this split transaction. Last was defined in section periodic interrupt - Do Complete Split. If it is the last complete-split (with a NYET response), the transfer state of the siTD is not advanced (never received any data) and the Active bit is cleared. No bits are set in the status field because this is essentially a skipped transaction. The transaction translator must have responded to all the scheduled complete-splits with NYETs, meaning the start-split issued by the host controller was not received. This result should be interpreted by system software as if the transaction was completely skipped. The test for whether this is the last complete split can be performed by XORing C-mask with C-prog-mask. A zero result indicates that all complete-splits have been executed.
- MDATA (and Last). See above description for testing for Last. This can only occur when there is an error condition. There has been a babble condition on the full-speed link, which delayed the completion of the full-speed transaction or software set up the S-mask and/or C-masks incorrectly. The host controller must set the XactErr bit and clear the active bit.
- NYET (and not Last). See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for C-prog-mask) and stay in this state.
- MDATA (and not Last). The transaction translator responds with an MDATA when it has partial data for the split transaction. For example, the full-speed transaction data payload spans from micro-frame X to X+1 and during micro-frame X, the transaction translator responds with an MDATA and the data accumulated up to the end of micro-frame X. The host controller advances the transfer state to reflect the number of bytes received.

If Test A succeeds, but Test B fails, it means one or more of the complete-splits have been skipped. The host controller sets the missed micro-frame status bit and clears the active bit.

Complete-Split for Scheduling Boundary Cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see Figure 32-68) require the host controller use the transaction state context of the previous siTD to finish the split transaction. Table 32-79 enumerates the transaction state fields.

Table 32-79. Summary siTD Split Transaction State

Buffer State	Status	Execution Progress
Total Bytes To Transfer P (page select) Current Offset TP (transaction position) T-count (transaction count)	All bits in the status field	C-prog-mask

NOTE

TP and T-count are only for host to device (OUT) endpoints.

If software has budgeted the schedule of this data stream with a frame wrap case, it must initialize the siTD [Back Pointer] field to reference a valid siTD and have the T bit in the siTD [Back Pointer] field cleared. Otherwise, software must set the T bit in siTD [Back Pointer]. The host controller's rules for interpreting when to use the siTD [Back Pointer] field are listed below. These rules apply only when the siTD's active bit is a one and the SplitXState is Do Complete Split.

- When cMicroFrameBit is a 0x1 and the $siTD_X$[Back Pointer] T-bit is zero, or
- If cMicroFrameBit is a 0x2 and siTDX[S-mask[0]] is zero

When either of these conditions apply, the host controller must use the transaction state from $siTD_{X-1}$.

To access $siTD_{X-1}$, the host controller reads on-chip the siTD referenced from $siTD_X$ [Back Pointer].

The host controller must save the entire state from $siTD_X$ while processing $siTD_{X-1}$. This is to accommodate for case 2b processing. The host controller must not recursively walk the list of siTD [Back Pointers].

If $siTD_{X-1}$ is active (Active bit is set and SplitXStat is Do Complete Split), Test A and Test B are applied as described above. If these criteria to execute a complete-split are met, the host controller executes the complete split and evaluates the results as described above. The transaction state (see Table 32-79) of $siTD_{X-1}$ is appropriately advanced based on the results and written back to memory. If the resultant state of $siTD_{X-1}$'s active bit is a one, the host controller returns to the context of $siTD_X$ and follows its next pointer to the next schedule item. No updates to $siTD_X$ are necessary.

If $siTD_{X-1}$ is active (Active bit is set and SplitXStat is Do Start Split), the host controller must clear the active bit and set the missed micro-frame status bit and the resultant status is written back to memory.

If $siTD_{X-1}$'s Active bit is cleared, (because it was cleared when the host controller first visited $siTD_{X-1}$ via $siTD_X$'s back pointer, it transitioned to zero as a result of a detected error, or the results of $siTD_{X-1}$'s complete-split transaction cleared it), the host controller returns to the context of $siTD_X$ and transitions its SplitXState to Do Start Split. The host controller then determines whether the case 2b start split boundary condition exists (that is, if cMicroframeBit is 1 and $siTD_X$[S-mask[0]] is 1). If this criterion is met, the host controller immediately executes a start-split transaction and appropriately advances the transaction state of $siTD_X$, then follows $siTD_X$[Next Pointer] to the next schedule item. If the criterion is not met, the host controller simply follows $siTD_X$[Next Pointer] to the next schedule item. In the case of a 2b boundary case, the split-transaction of $siTD_{X-1}$ has its active bit cleared when the host controller returns to the context of $siTD_X$. Software should not initialize an siTD with C-mask bits 0 and 1 set and an S-mask with bit 0 set. This scheduling combination is not supported and the behavior of the host controller is undefined.

32.6.11.3.4 Split Transaction for Isochronous - Processing Examples

There is an important difference between how the hardware/software manages the isochronous split transaction state machine and how it manages the asynchronous and interrupt split transaction state machines. The asynchronous and interrupt split transaction state machines are encapsulated within a single queue head. The progress of the data stream depends on the progress of each split transaction. In some respects, the split-transaction state machine is sequenced using the execute transaction queue head traversal state machine.

Isochronous is a pure time-oriented transaction/data stream. The interface data structures are optimized to efficiently describe transactions that need to occur at specific times. The isochronous split-transaction state

machine must be managed across these time-oriented data structures. This means that system software must correctly describe the scheduling of split-transactions across more than one data structure.

The host controller must make the appropriate state transitions at the appropriate times, in the correct data structures.

For example, Table 32-80 illustrates a few frames worth of scheduling required to schedule a case 2a full-speed isochronous data stream.

Table 32-80. Example Case 2a - Software Scheduling siTDs for an IN Endpoint

siTDX		Micro-Frames								InitialSplitXState
#	Masks	0	1	2	3	4	5	6	7	
X	S-Mask					1				Do Start Split
	C-Mask	1	1					1	1	
X + 1	S-Mask					1				Do Complete Split
	C-Mask	1	1					1	1	
X + 2	S-Mask					1				Do Complete Split
	C-Mask	1	1					1	1	
X + 3	S-Mask	Repeats previous pattern								Do Complete Split
	C-Mask									

This example shows the first three siTDs for the transaction stream. Because this is the case-2a frame-wrap case, S-masks of all siTDs for this endpoint have a value of 0x10 (a one bit in micro-frame 4) and C-mask value of 0xC3 (one-bits in micro-frames 0,1, 6 and 7). Additionally, software ensures that the back pointer field of each siTD references the appropriate siTD data structure (and the Back Pointer T-bits are cleared).

The initial SplitXState of the first siTD is Do Start Split. The host controller visits the first siTD eight times during frame X. The C-mask bits in micro-frames 0 and 1 are ignored because the state is Do Start Split. During micro-frame 4, the host controller determines it can run a start-split (and does) and changes SplitXState to Do Complete Split. During micro-frames 6 and 7, the host controller executes complete-splits. The siTD for frame X+1 has its SplitXState initialized to Do Complete Split. As the host controller continues to traverse the schedule during H-Frame X+1, it visits the second siTD eight times. During micro-frames 0 and 1, it detects that it must execute complete-splits.

During H-Frame X+1, micro-frame 0, the host controller detects that $siTD_{X+1}$'s back pointer [T] bit is a zero, saves the state of $siTD_{X+1,}$ and fetches $siTD_X$. It executes the complete split transaction using the transaction state of $siTD_X$. If the $siTD_X$ split transaction is complete, siTD's active bit is cleared and results written back to $siTD_X$. The host controller retains the fact that $siTD_X$ is retired and transitions the SplitXState in $siTD_{X+1}$ to Do Start Split. At this point, the host controller is prepared to execute the start-split for $siTD_{X+1}$ when it reaches micro-frame 4. If the split-transaction completes early (transaction-complete is defined in Periodic Isochronous - Do Complete Split), before all the scheduled complete-splits have been executed, the host controller changes $siTD_X$[SplitXState] to Do Start Split early and naturally skips the remaining scheduled complete-split transactions. For this example, $siTD_{X+1}$ does not receive a DATA0 response until H-Frame X+2, micro-frame 1.

During H-Frame X+2, micro-frame 0, the host controller detects that $siTD_{X+2}$'s back pointer [T] bit is zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. As described above, it executes another split transaction, receives an MDATA response, updates the transfer state, but does not modify the active bit. The host controller returns to the context of $siTD_{X+2}$, and traverses it's next pointer without any state change updates to $siTD_{X+2}$.

During H-Frame X+2, micro-frame 1, the host controller detects $siTD_{X+2}$'s S-mask[0] bit is zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. It executes another complete-split transaction, receives a DATA0 response, updates the transfer state and clears the Active bit. It returns to the state of $siTD_{X+2}$ and changes its SplitXState to Do Start Split. At this point, the host controller is prepared to execute start-splits for $siTD_{X+2}$ when it reaches micro-frame 4.

32.6.12 Host Controller Pause

When the host controller's HCHalted bit in the USB_USBSTS register is a zero, the host controller is sending SOF (Start OF Frame) packets down all enabled ports. When the schedules are enabled, the EHCI host controller accesses the schedules in main memory each micro-frame. This constant pinging of main memory is known to create CPU power management problems for mobile systems. Specifically, mobile systems aggressively manage the state of the CPU, based on recent history usage. In the more aggressive power saving modes, the CPU can disable its caches. Current PC architectures assume that bus-master accesses to main memory must be cache-coherent. So, when bus masters are busy touching memory, the CPU power management software can detect this activity over time and inhibit the transition of the CPU into its lowest power savings mode. USB controllers are bus-masters and the frequency at which they access their memory-based schedules keeps the CPU power management software from placing the CPU into its lowest power savings state.

USB Host controllers don't access main memory when they are suspended. However, there are a variety of reasons why placing the USB controllers into suspend won't work, but they are beyond the scope of this document. The base requirement is that the USB controller needs to be kept out of main memory, while at the same time, the USB bus is kept from going into suspend.

EHCI controllers provide a large-grained mechanism that can be manipulated by system software to change the memory access pattern of the host controller. System software can manipulate the schedule enable bits in the USB_USBCMD register to turn on/off the scheduling traversal. A software heuristic can be applied to implement an on/off duty cycle that allows the USB to make reasonable progress and allow the CPU power management to get the CPU into its lowest power state. This method is not intended to be applied at all times to throttle USB, but should only be applied in specific configurations and usage loads. For example, when only a keyboard or mouse is attached to the USB, the heuristic could detect times when the USB is attempting to move data only infrequently and can adjust the duty cycle to allow the CPU to reach its low power state for longer periods of time. Similarly, it could detect increases in the USB load and adjust the duty cycle appropriately, even to the point where the schedules are never disabled. The assumption here is that the USB is moving data and the CPU is required to process the data streams.

To provide a complete solution for the system, the companion host controllers should also provide a similar method to allow system software to inhibit the companion host controller from accessing it's shared memory based data structures (schedule lists or otherwise).

32.6.13 Port Test Modes

EHCI host controllers implement the port test modes Test J_State, Test K_State, Test_Packet, Test Force_Enable, and Test SE0_NAK as described in the *USB Specification Revision 2.0*. The required, port test sequence is (assuming the CF bit in the USB_CONFIGFLAG register is set):

- Disable the periodic and asynchronous schedules by clearing the asynchronous schedule enable and periodic schedule enable bits in the USB_USBCMD register.
- Place all enabled root ports into the suspended state by setting the suspend bit in each appropriate USB_PORTSC*n* register.
- Clear the run/stop bit in the USB_USBCMD register and wait for the HCHalted bit in the USB_USBSTS register, to transition to a 1. An EHCI host controller implementation may optionally allow port testing with the run/stop bit set. However, all host controllers must support port testing with run/stop cleared and HCHalted set.
- Set the port test control field in the port under test USB_PORTSC*n* register to the value corresponding to the desired test mode. If the selected test is Test_Force_Enable, the run/stop bit in the USB_USBCMD register must be transitioned back to one to enable transmission of SOFs out of the port under test.
- When the test is complete, system software must ensure the host controller is halted (HCHalted bit = 1), it terminates and exits test mode by setting HCReset.

32.6.14 Interrupts

The EHCI host controller hardware provides interrupt capability based on a number of sources. There are several general groups of interrupt sources:

- Interrupts as a result of executing transactions from the schedule (success and error conditions),
- Host controller events (Port change events, etc.), and
- Host controller error events

All transaction-based sources are maskable through the host controller's interrupt enable register (USB_USBINTR). Additionally, individual transfer descriptors can be marked to generate an interrupt on completion. This section describes each interrupt source and the processing that occurs in response to the interrupt.

During normal operation, interrupts may be immediate or deferred until the next interrupt threshold occurs. The interrupt threshold is a tunable parameter via the interrupt threshold control field in the USB_USBCMD register. The value of this register controls when the host controller generates an interrupt on behalf of normal transaction execution. When a transaction completes during an interrupt interval period, the interrupt signaling the completion of the transfer does not occur until the interrupt threshold occurs. For example, the default value is eight micro-frames. This means that the host controller does not generate interrupts any more frequently than once every eight micro-frames.

Section 32.6.14.2.4, “Host System Error,” details the effects of a host system error.

If an interrupt is scheduled to be generated for the current interrupt threshold interval, the interrupt is not signaled until after the status for the last complete transaction in the interval has been written back to system memory. This may result in the interrupt not being signaled until the next interrupt threshold.

Initial interrupt processing is the same, regardless of the reason for the interrupt. When an interrupt is signaled by the hardware, CPU control is transferred to host controller's USB interrupt handler. The precise mechanism to accomplish the transfer is OS specific. For this discussion, it is assumed control is received. When the interrupt handler receives control, its first action is to read the USB_USBSTS. It then acknowledges the interrupt by clearing all of the interrupt status bits by writing ones to these bit positions. The handler then determines whether the interrupt is due to schedule processing or some other event. After acknowledging the interrupt, the handler (via an OS-specific mechanism) schedules a deferred procedure call (DPC) that executes later. The DPC routine processes the results of the schedule execution. The precise mechanisms used are beyond the scope of this document.

NOTE

The only method software should use for acknowledging an interrupt is by transitioning the appropriate status bits in the USB_USBSTS register from a 1 to a 0.

32.6.14.1 Transfer/Transaction Based Interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

32.6.14.1.1 Transaction Error

A transaction error is any error that caused the host controller to think the transfer did not complete successfully. Table 32-81 lists the events/responses the host can observe as a result of a transaction. The effects of the error counter and interrupt status are summarized in the following paragraphs. Most of these errors set the XactErr status bit in the appropriate interface data structure.

Table 32-81. Summary of Transaction Errors

Event/ Result	Queue Head/qTD/iTD/siTD Side Effects		USB_USBSTS[USBERRINT]
	Cerr	Status Field	
CRC	–1	XactErr set	1[1]
Timeout	–1	XactErr set	1[1]
Bad PID[2]	–1	XactErr set	1[1]
Babble	N/A	See Serial Bus Babble	1
Buffer Error	N/A	See Data Buffer Error	

1 If occurs in a queue head, USBERRINT is asserted only when Cerr counts down from a 1 to a 0. In addition, the queue is halted.

2 The host controller received a response from the device, but it could not recognize the PID as a valid PID.

There is a small set of protocol errors that relate only when executing a queue head and fit under the umbrella of a WRONG PID error that are significant to explicitly identify. When these errors occur, the XactErr status bit in the queue head is set and the Cerr field is decremented. When the PID Code indicates a SETUP, the following responses are protocol errors and result in XactErr bit being set and the Cerr field being decremented.

- EPS field indicates a high-speed device and it returns a Nak handshake to a SETUP.
- EPS field indicates a high-speed device and it returns a Nyet handshake to a SETUP.
- EPS field indicates a low- or full-speed device and the complete-split receives a Nak handshake.

Serial Bus Babble

When a device transmits more data on the USB than the host controller is expecting for this transaction, it is defined to be babbling. In general, this is called a packet babble. When a device sends more data than the maximum length number of bytes, the host controller sets the babble detected bit to 1 and halts the endpoint if it is using a queue head. Maximum length is defined as the minimum of total bytes to transfer and maximum packet size. The Cerr field is not decremented for a packet babble condition (only applies to queue heads). A babble condition also exists if IN transaction is in progress at high-speed EOF2 point. This is called a frame babble. A frame babble condition is recorded into the appropriate schedule data structure. In addition, the host controller must disable the port to which the frame babble is detected.

The USBERRINT bit in the USB_USBSTS register is set. If the USB error interrupt enable bit in the USB_USBINTR register is set, a hardware interrupt is signaled to the system at the next interrupt threshold. The host controller must never start an OUT transaction that babbles across a micro-frame EOF.

When a host controller detects a data PID mismatch, it must disable the packet babble checking for the duration of the bus transaction or do packet babble checking based solely on maximum packet size. The USB core specification defines the requirements on a data receiver when it receives a data PID mismatch (for example, expects a DATA0 and gets a DATA1 or visa-versa). In summary, it must ignore the received data and respond with an ACK handshake to advance the transmitter's data sequence.The EHCI interface allows system software to provide buffers for a control, bulk, or interrupt IN endpoint that are not an even multiple of the maximum packet size specified by the device. When a device misses an ACK for an IN endpoint, the host and device are out of synchronization with respect to the progress of the data transfer. The host controller may have advanced the transfer to a buffer that is less than maximum packet size. The device re-sends its maximum packet size data packet, with the original data PID, in response to the next IN token. To properly manage the bus protocol, the host controller must disable the packet babble check when it observes the data PID mismatch.

Data Buffer Error

This event indicates that either an overrun of incoming data or an underrun of outgoing data has occurred for this transaction. This would generally be caused by the host controller not being able to access required data buffers in memory within necessary latency requirements. These conditions are not considered transaction errors and do not affect the error count in the queue head. When these errors do occur, the host controller records the fact the error occurred by setting the data buffer error bit in the queue head, iTD or siTD.

If the data buffer error occurs on a non-isochronous IN, the host controller does not issue a handshake to the endpoint. This forces the endpoint to resend the same data (and data toggle) in response to the next IN to the endpoint.

If the data buffer error occurs on an OUT, the host controller must corrupt the end of the packet so that it cannot be interpreted by the device as a good data packet. Simply truncating the packet is not considered acceptable. An acceptable implementation option is to one's complement the CRC bytes and send them.

There are other options suggested in the Transaction Translator section of the *USB Specification Revision 2.0*.

32.6.14.1.2 USB Interrupt (Interrupt on Completion (IOC))

Transfer descriptors (iTDs, siTDs, and queue heads (qTDs)) contain a bit that can cause an interrupt on their completion. The completion of the transfer associated with that schedule item causes the USB Interrupt (USBINT) bit in the USB_USBSTS register to be set. In addition, if a short packet is encountered on an IN transaction associated with a queue head, this event also causes USBINT to be set. If the USB interrupt enable bit in the USB_USBINTR register is set, a hardware interrupt is signaled to the system at the next interrupt threshold. If the completion is because of errors, the USBERRINT bit in the USB_USBSTS register is also set.

32.6.14.1.3 Short Packet

Reception of a data packet less than the endpoint's max packet size during control, bulk, or interrupt transfers signals the completion of the transfer. When a short packet completion occurs during a queue head execution, the USBINT bit in the USB_USBSTS register is set. If the USB interrupt enable bit is set in the USB_USBINTR register, a hardware interrupt is signaled to the system at the next interrupt threshold.

32.6.14.2 Host Controller Event Interrupts

These interrupt sources are independent of the interrupt threshold, with the one exception being the interrupt on async advance.

32.6.14.2.1 Port Change Events

Port registers contain status and status change bits. When the status change bits are set, the host controller sets the port change detect bit in the USB_USBSTS register. If the port change interrupt enable bit in the USB_USBINTR register is set, the host controller issues a hardware interrupt. The port status change bits are:

- Connect Status Change
- Port Enable/Disable Change
- Over-current Change
- Force Port Resume

32.6.14.2.2 Frame List Rollover

This event indicates the host controller has wrapped the frame list. The current programmed size of the frame list affects how often this interrupt occurs. (If the frame list size is 1024, the interrupt occurs every 1024 milliseconds. If it is 512, it occurs every 512 milliseconds, etc.) When a frame list rollover is detected, the host controller sets the frame list rollover bit in the USB_USBSTS register. If the frame list rollover enable bit in the USB_USBINTR register is set, the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

32.6.14.2.3 Interrupt on Async Advance

This event is used for deterministic removal of queue heads from the asynchronous schedule. When the host controller advances the on-chip context of the asynchronous schedule, it evaluates the value of the interrupt on async advance doorbell bit in the USB_USBCMD register. If it is set, it sets the interrupt on async advance bit in the USB_USBSTS register. If the interrupt on async advance enable bit in the USB_USBINTR register is set, the host controller issues a hardware interrupt at the next interrupt threshold. A detailed explanation of this feature is described in Section 32.6.7.2, “Removing Queue Heads from Asynchronous Schedule.”

32.6.14.2.4 Host System Error

The host controller is a bus master and any interaction between the host controller and the system may experience errors. The type of host error may be catastrophic to the host controller making it impossible for the host controller to continue in a coherent fashion. Behavior for these types of errors is to halt the host controller. Host-based error must result in the following actions:

- The run/stop bit in the USB_USBCMD register is cleared.
- The host system error and HCHalted bits in the USB_USBSTS register are set.
- If the host system error enable bit in the USB_USBINTR register is set, the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

Table 32-82 summarizes the required actions taken on the various host errors.

Table 32-82. Summary Behavior on Host System Errors

Cycle Type	Master Abort	Target Abort	Data Phase Parity
Frame list pointer fetch (read)	Fatal	Fatal	Fatal
siTD fetch (read)	Fatal	Fatal	Fatal
siTD status write-back (write)	Fatal	Fatal	Fatal
iTD fetch (read)	Fatal	Fatal	Fatal
iTD status write-back (write)	Fatal	Fatal	Fatal
qTD fetch (read)	Fatal	Fatal	Fatal
qHD status write-back (write)	Fatal	Fatal	Fatal
Data write	Fatal	Fatal	Fatal
Data read	Fatal	Fatal	Fatal

NOTE

After a host system error, software must reset the host controller using HCReset in the USB_USBCMD register before re-initializing and restarting the host controller.

32.7 Device Data Structures

This section defines the interface data structures that communicate control, status, and data between device controller driver (DCD) software and the device controller. Data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of device queue heads and transfer descriptors.

NOTE

Software must ensure no interface data structure reachable by the device controller spans a 4K-page boundary.

The data structures defined in this section are (from the device controller's perspective) a mix of read-only and read/writable fields. The device controller must preserve the read-only fields on all data structure writes.

The USB core includes DCD software called the USB 2.0 device API. The device API provides an easy to use application program interface for developing device (peripheral) applications. The device API incorporates and abstracts for the application developer all of the elements of the program interface.

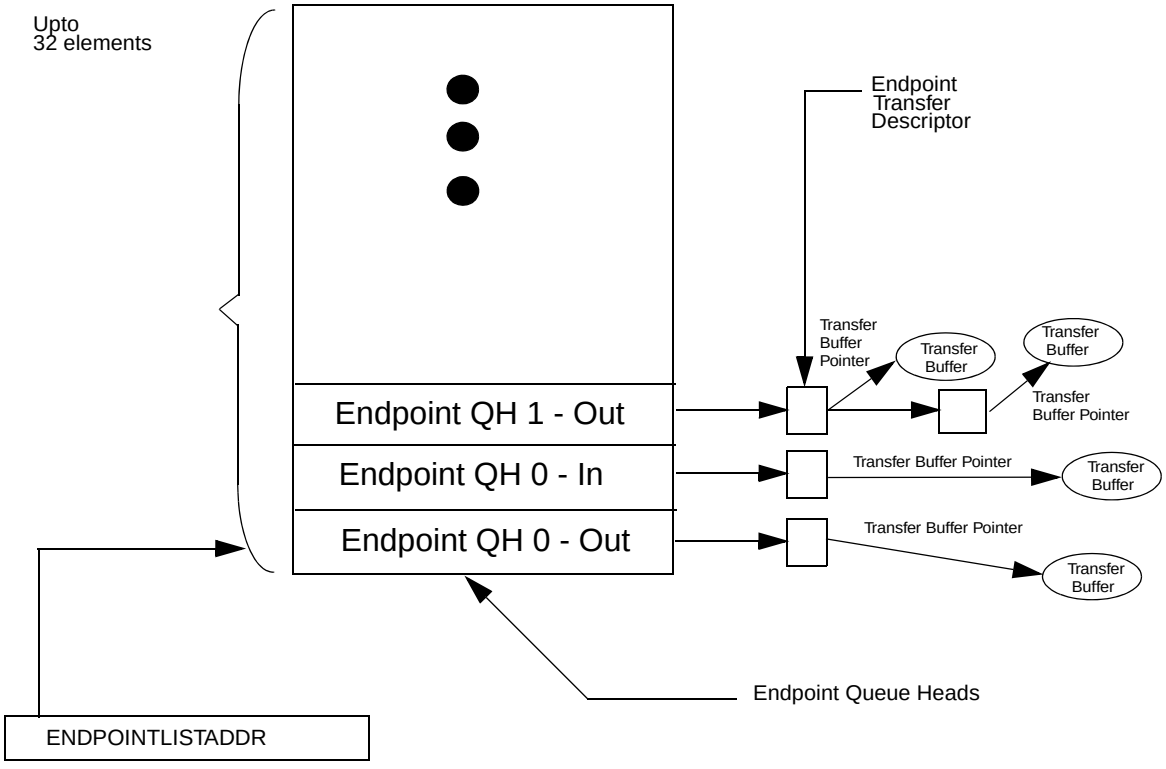


Figure 32-71. End Point Queue Head Organization

32.7.1 Endpoint Queue Head

The device Endpoint Queue Head (dQH) is where all transfers are managed. The dQH is a 48-byte data structure, but must be aligned on 64-byte boundaries. During priming of an endpoint, the dTD (device

transfer descriptor) is copied into the overlay area of the dQH, which starts at the next TD pointer 32-bit word and continues through the end of the buffer pointers 32-bit words. After a transfer is complete, the dTD status 32-bit word is updated in the dTD pointed to by the current TD pointer. While a packet is in progress, the overlay area of the dQH is a staging area for the dTD so the device controller can access needed information with little minimal latency.

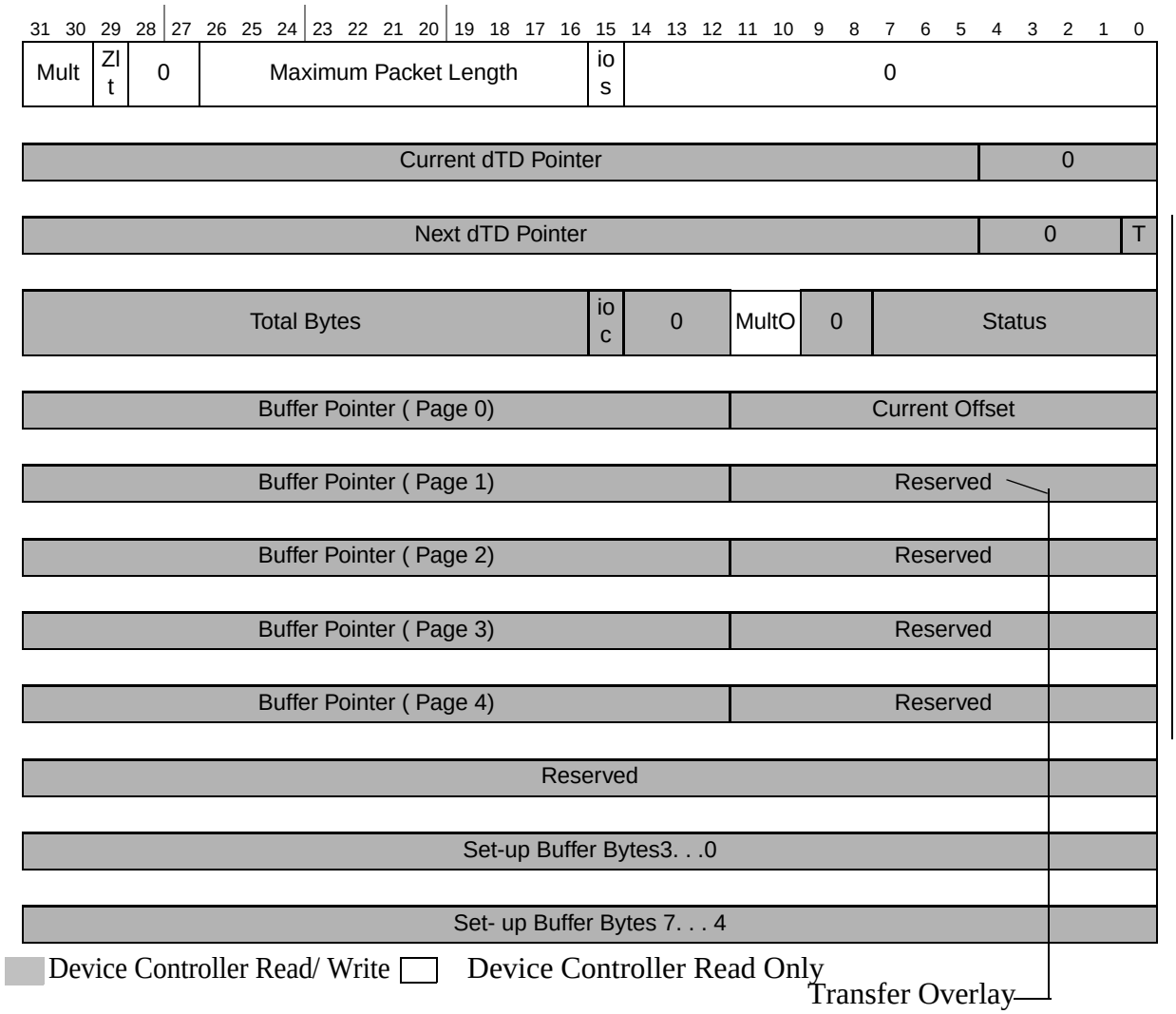


Figure 32-72. Endpoint Queue Head

32.7.1.1 Endpoint Capabilities/Characteristics

This 32-bit word specifies static information about the endpoint; in other words, this information does not change over the lifetime of the endpoint. Device controller software should not attempt to modify this information while the corresponding endpoint is enabled.

Table 32-83. Endpoint Capabilities/Characteristics

Bit	Description
31:30	Mult. This field indicates the number of packets executed per transaction description as given by the following: 00 Execute n transactions as demonstrated by the USB variable length packet protocol where n is computed using the maximum packet length (dQH) and the total bytes field (dTD) 01 Execute 1 Transaction 10 Execute 2 Transactions 11 Execute 3 Transactions **Note:** Non-ISO endpoints must set Mult equal to 00. **Note:** ISO endpoints must set Mult equal to 01, 10, or 11 as needed.
29	Zero Length Termination Select. This bit indicates when a zero length packet terminates transfers where total transfer length is a multiple. This bit is not relevant for isochronous transfers. 0 Enable zero length packet to terminate transfers equal to a multiple of the maximum packet length. (default). 1 Disable the zero length packet on transfers equal in length to a multiple maximum packet length.
28:27	Reserved. These bit reserved for future use and should be set to 0.
26:16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).
15	Interrupt On Setup (IOS). This bit is used on control type endpoints to indicate if USBINT is set in response to a setup being received.
14:0	Reserved. Bits reserved for future use and should be set to 0.

32.7.1.2 Transfer Overlay

The seven 32-bit words in the overlay area represent a transaction working space for the device controller. The general operational model is that the device controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, it does not read the associated endpoint.

After an endpoint is readied, the dTD is copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller writes the results back to the original transfer descriptor and advances the queue.

See dTD for a description of the overlay fields.

32.7.1.3 Current dTD Pointer

The device controller uses the current dTD pointer to locate the transfer in progress. This word is for hardware use only, and DCD software should not modified it.

Table 32-84. Current dTD Pointer

Bit	Description
31:5	Current dTD. This field is a pointer to the dTD that is represented in the transfer overlay area. This field is modified by the device controller to the next dTD pointer during endpoint priming or queue advance.
4:0	Reserved. Bit reserved for future use and should be set to zero.

32.7.1.4 Set-up Buffer

The set-up buffer is dedicated storage for the 8-byte data that follows a set-up PID.

NOTE

Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is for receiving setup data packets.

Table 32-85. Multiple Mode Control

32-bit word	Bits	Description
1	31:0	Setup Buffer 0. This buffer contains bytes 3 to 0 of an incoming setup buffer packet and the device controller writes it for software to read.
2	31:0	Setup Buffer 1. This buffer contains bytes 7 to 4 of an incoming setup buffer packet and the device controller writes it for software to read.

32.7.2 Endpoint Transfer Descriptor (dTD)

The dTD describes to the device controller the location and quantity of data sent/received for given transfer. The DCD should not attempt to modify any field in an active dTD except the next link pointer, which should only be modified as described in Section 32.8.7, “Managing Transfers with Transfer Descriptors.”

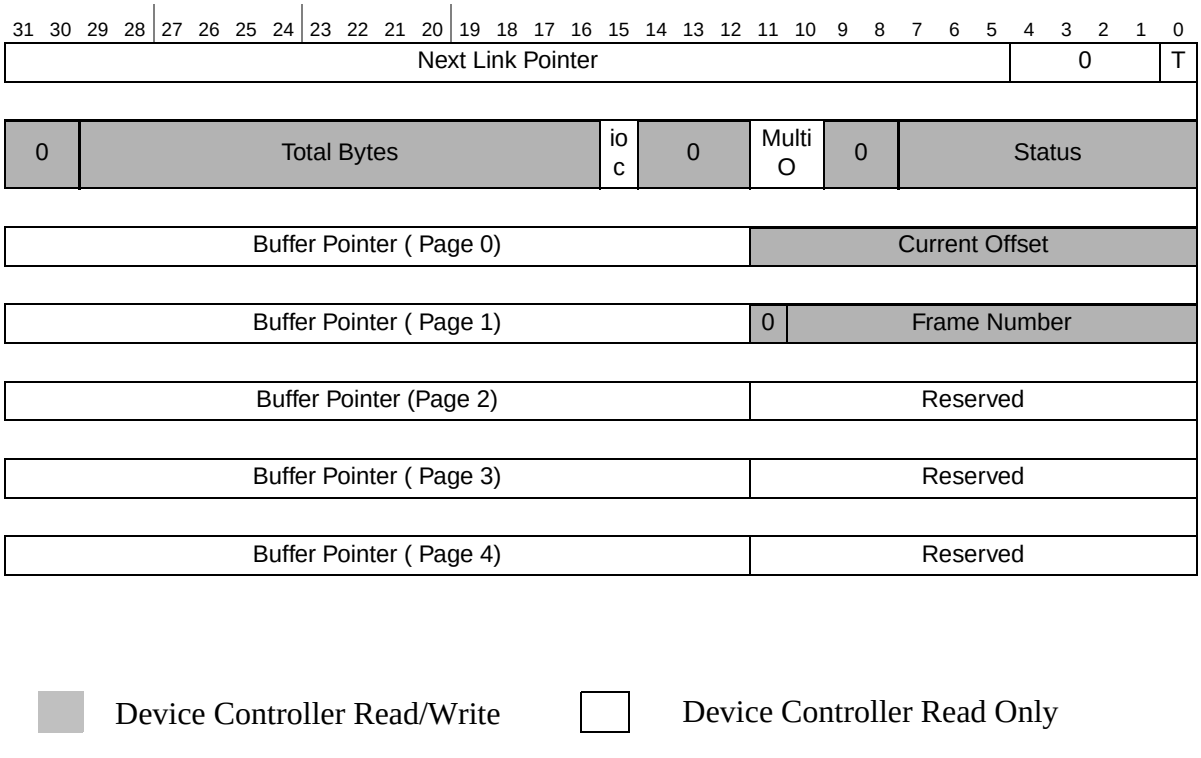


Figure 32-73. Endpoint Transfer Descriptor (dTD)

Table 32-86. Next dTD Pointer

Bit	Description
31:5	Next Transfer Element Pointer. This field contains the physical memory address of the next dTD to be processed. The field corresponds to memory address signals [31:5], respectively.
4:1	Reserved. Bits reserved for future use and should be set to 0.
0	Terminate (T). 1=pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the device controller that no more valid entries exist in the queue.

Table 32-87. dTD Token

Bit	Description
31	Reserved. Bit reserved for future use and should be set to 0.
30:16	Total Bytes. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction and only on the successful completion of the transaction. The maximum value software may store in the field is 5 × 4 KB (0x5000). This is the maximum number of bytes 5 page pointers can access. Although it is possible to create a transfer as large as 20 KB this assumes the first offset into the first page is 0. When the offset cannot be predetermined, crossing passed the fifth page can be guaranteed by limiting the total bytes to 16 KB. Therefore, the maximum recommended transfer is 16 KB (0x4000). If the value of the field is 0 when the host controller fetches this transfer descriptor (and the active bit is set), the device controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for IN transfers that total bytes to transfer be an even multiple of maximum packet length. If software builds such a transfer descriptor for an IN transfer, the last transaction is always less that maximum packet length.
15	Interrupt On Complete (IOC). This bit indicates if USBINT is to be set in response to device controller being finished with this dTD.
14:12	Reserved. Bits reserved for future use and should be set to 0.
11:10	Multiplier Override (MultO). This field can be used for transmit ISO's (i.e., ISO-IN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit-ISO. Example: if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 0 [default] Three packets are sent: {Data2(8); Data1(7); Data0(0)} if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 2 Two packets are sent: {Data1(8); Data0(7)} For maximal efficiency, software should compute MultO = greatest integer of (Total Bytes/Max. Packet Size) except for the case when Total Bytes = 0; then MultO should be 1. **Note:** Non-ISO and Non-TX endpoints must set MultO = 00.

Table 32-87. dTD Token (continued)

Bit	Description
9:8	Reserved. Bits reserved for future use and should be set to zero.
7:0	Status. This field is used by the Device Controller to communicate individual command execution states back to the Device Controller software. This field contains the status of the last transaction performed on this qTD. The bit encodings are: Bit Status Field Description 7 Active. 6 Halted. 5 Data Buffer Error. 3 Transaction Error. 4,2,0Reserved.

Table 32-88. Buffer Page Pointer List

Bit	Description
31:12	Buffer Pointer. Selects the page offset in memory for the packet buffer. Non-virtual memory systems typically set the buffer pointers to a series of incrementing integers.
0;11:0	Current Offset. Offset into the 4kb buffer where the packet is to begin.
1;10:0	Frame Number. Written by the device controller to indicate the frame number in which a packet finishes. This is typically used to correlate relative completion times of packets on an ISO endpoint.

32.8 Device Operational Model

The device operation transfers a request in the memory image to and from the Universal Serial Bus. Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller performs the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point-of-view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

32.8.1 Device Controller Initialization

After hardware reset, the USB module is disabled until the run/stop bit is set to 1. In the disabled state, the pull-up on the USB D+ is not active, which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs. Shortly after the device is enabled, a USB reset occurs followed by setup packet arriving at endpoint 0. A queue head must be prepared so the device controller can store the incoming setup packet.

To initialize a device, software must perform these steps:

1. Set controller mode to device mode. Optionally set streaming disable in the USB_USBMODE register.

NOTE

Transitioning from host mode to device mode requires a device controller reset before modifying USB_USBMODE.

2. Optionally modify the USB_BURSTSIZE register.

3. Program the PTS field of the USB_PORTSC*n* register if using a non-ULPI PHY.
4. Set iodis_b bit in the CONTROL register to enable PHY interface.
5. Allocate and Initialize device queue heads in system memory Minimum: Initialize device queue heads 0 Tx and 0 Rx.

NOTE

All device queue heads must be initialized for control endpoints before the endpoint is enabled. Device queue heads for non-control endpoints must be initialized before the endpoint can be used.

For information on device queue heads, refer to Section 32.7, "Device Data Structures."

6. Configure USB_ENDPOINTLISTADDR Pointer.

   For additional information on USB_ENDPOINTLISTADDR, refer to the register table.
7. Enable the microprocessor interrupt associated with the USB controller and optionally change setting of ITC field in USB_USBCMD register.

   Recommended: Enable all device interrupts including: USBINT, USBERRINT, port change detect, USB reset received, and DCSuspend.

   For a list of available interrupts, refer to the USB_USBINTR and the USB_USBSTS register tables.
8. Set run/stop bit to run mode.

   After the run bit is set, a device reset occurs. The DCD must monitor the reset event and set the USB_DEVICEADDR register, set the USB_ENDPTCTRL*n* registers, and adjust the software state as described in the bus reset section of the following port state and control section below.

NOTE

Endpoint 0 is a control endpoint only and does not need to be configured using USB_ENDPTCTRL0 register.

It is also not necessary to initially prime Endpoint 0 because the first packet received is always a setup packet. The contents of the first setup packet requires a response in accordance with USB device framework command set.

32.8.2 Port State and Control

From a chip or system reset, the USB controller enters the *powered* state. A transition from the powered state to the *attach* state occurs when the run/stop bit is set to 1. After receiving a reset on the bus, the port enters the *defaultFS or defaultHS* state in accordance with the protocol reset described in Appendix C.2 of the USB Specification Rev. 2.0. The following state diagram (Figure 32-74) depicts the state of a USB 2.0 device.

States powered, attach, defaultFS/HS, and suspendFS/HS are implemented in the USB controller and are communicated to the DCD using these bits:

Table 32-89. Device Controller State Information Bits

Bit	Register
DCSuspend	USB_USBSTS
USB Reset Received	USB_USBSTS
Port Change Detect	USB_USBSTS
High-Speed Port	USB_PORTSC*n*

It is the responsibility of the DCD to maintain a state variable to differentiate between the defaultFS/HS state and the address/configured states. Change of state from default to address and the configured states is part of the enumeration process described in the device framework section of the USB 2.0 Specification.

As a result of entering the address state, the DCD must program the USB_DEVICEADDR register.

Entry into the Configured indicates all endpoints are used in the operation of the device have been properly initialized by programming the USB_ENDPTCTRL*n* registers and initializing the associated queue heads.

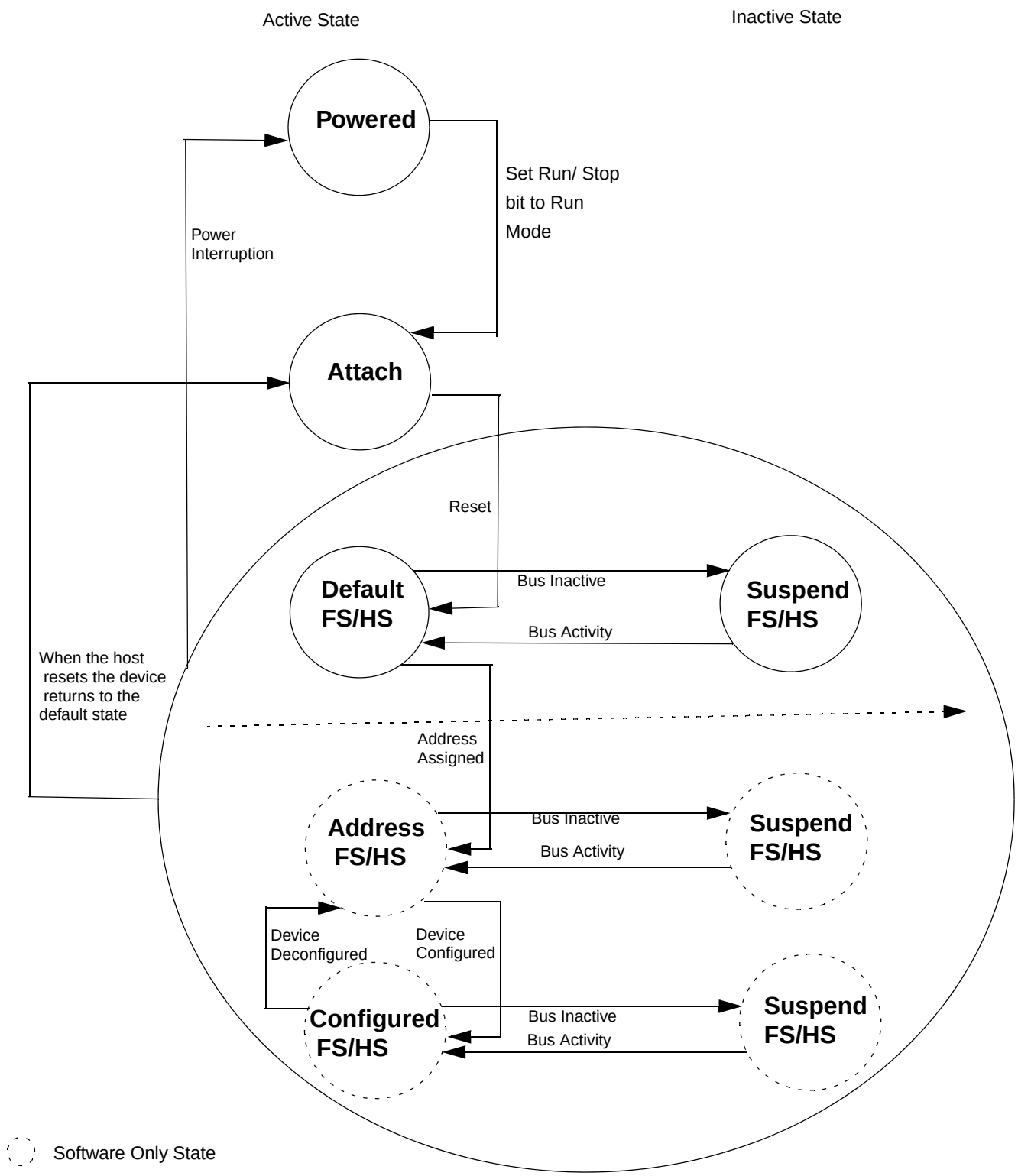


Figure 32-74. USB 2.0 Device States

32.8.3 Bus Reset

The host uses a bus reset to initialize downstream devices. When a bus reset is detected, the USB controller renegotiates its attachment speed, resets the device address to 0, and notifies the DCD by interrupt (assuming the USB reset interrupt enable is set). After a reset is received, all endpoints (except endpoint 0) are disabled and the device controller cancels any primed transactions. The concept of priming is clarified below, but the DCD must perform the following tasks when a reset is received:

- Clear all setup token semaphores by reading the USB_ENDPTSETUPSTAT register and writing the same value back to the USB_ENDPTSETUPSTAT register.
- Clear all the endpoint complete status bits by reading the USB_ENDPTCOMPLETE register and writing the same value back to the USB_ENDPTCOMPLETE register.
- Cancel all primed status by waiting until all bits in the USB_ENDPTPRIME are 0 and then writing 0xFFFF_FFFF to USB_ENDPTFLUSH.

Read the reset bit in the USB_PORTSC*n* register and make sure that remains active. A USB reset occurs for a minimum of three ms and the DCD must reach this point in the reset cleanup before end of the reset occurs; otherwise, a hardware reset of the device controller is recommended (rare.)

Writing a 1 to the USB controller reset bit in the USB_USBCMD reset can perform a hardware reset.

NOTE

A hardware reset causes the device to detach from the bus by clearing the run/stop bit. Therefore, the DCD must completely re-initialize the USB controller after a hardware reset.

Free all allocated dTDs because they are no longer executed by the device controller. If this is the first time the DCD is processing a USB reset event, it is likely that no dTDs have been allocated.

At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a port change detect is indicated.

After a port change detect, the device has reached the default state and the DCD can read the USB_PORTSC*n* to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration.

NOTE

The device DCD may use the FS/HS mode information to determine the bandwidth mode of the device.

In some applications, it may not be possible to enable one or more pipes while in FS mode. Beyond the data rate issue, there is no difference in DCD operation between FS and HS modes.

32.8.3.1 Suspend/Resume

32.8.3.1.1 Suspend Description

To conserve power, the USB controller automatically enters the suspended state when no bus traffic has been observed for a specified period. When suspended, the USB controller maintains any internal status, including its address and configuration. In device mode, the attached devices must be prepared to suspend

at any time they are powered, regardless if they have been assigned a non-default address, are configured, or neither. Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the suspended state when the hub port it is attached to is disabled.

The USB controller exits suspend mode when there is bus activity. It may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wake-up. The ability of a device to signal remote wake-up is optional. The USB controller is capable of remote wake-up signaling. When the USB controller is reset, remote wake-up signaling must be disabled.

32.8.3.1.2 Suspend Operational Model

The USB controller moves into the suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period. After the device controller enters the suspend state, DCD is notified by an interrupt (assuming DC suspend interrupt is enabled). When the DCSuspend bit in the USB_PORTSC*n* is set to a 1, the device controller is suspended.

DCD response when the device controller is suspended is application specific and may involve switching to low-power operation.

Find information on the bus power limits in suspend state in USB 2.0 specification.

32.8.3.1.3 Resume

If the USB controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port. In addition, the USB controller can signal the system to resume operation by forcing resume signaling to the upstream port. Resume signaling is sent upstream by writing a '1' to the Resume bit in the in the USB_PORTSC*n* while the device is in suspend state. Sending resume signal to an upstream port should cause the host to issue resume signaling and bring the suspended bus segment (one more devices) back to the active condition.

NOTE

Before resume signaling can be used, the host must enable it by using the set feature command defined in device framework (chapter 9) of the USB 2.0 specification.

32.8.4 Managing Endpoints

The USB 2.0 specification defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device. The endpoint address is specified by the combination of the endpoint number and the endpoint direction.

The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a control type data channel used for device discovery and enumeration. Other types of endpoints supported by USB include bulk, interrupt, and isochronous. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the USB 2.0 specification.

Both USB controllers support as many as four bidirectional endpoints, including the control endpoint. The DCD can enable, disable, and configure each endpoint.

Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1-IN to be a bulk endpoint and endpoint 1-OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation. The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0 is, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a queue head allocated in memory. If the maximum of four endpoint numbers, one for each endpoint direction the device controller uses, eight queue heads are required. The operation of an endpoint and use of queue heads are described later in this document.

32.8.4.1 Endpoint Initialization

After hardware reset, all endpoints except endpoint zero are uninitialized and disabled. The DCD must configure and enable each endpoint by writing to configuration bit in the USB_ENDPTCTRL*n* register. Each 32-bit USB_ENDPTCTRL*n* is split into an upper and lower half. The lower half of USB_ENDPTCTRL*n* configures the receive or OUT endpoint and the upper half also configures the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the USB_ENDPTCTRL*n* register otherwise the behavior is undefined. The following table shows how to construct a configuration word for endpoint initialization.

Table 32-90. Device Controller Endpoint Initialization

Field	Value
Data Toggle Reset	1
Data Toggle Inhibit	0
Endpoint Type	00 — Control 01 — Isochronous 10 — Bulk 11 — Interrupt
Endpoint Stall	0

32.8.4.2 Stalling

There are two occasions where the USB controller may need to return to the host a STALL condition

The first occasion is the functional stall, which is a condition set by the DCD as described in the USB 2.0 device framework (chapter 9). A functional stall is used only on non-control endpoints and can be enabled in the device controller by setting the endpoint stall bit in the USB_ENDPTCTRL*n* register associated with the given endpoint and the given direction. In a functional stall condition, the device controller continues to return STALL responses to all transactions occurring on the respective endpoint and direction until the endpoint stall bit is cleared by the DCD.

A protocol stall, unlike a function stall, used on control endpoints is automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD

should enable the stall bits (both directions) as a pair. A single write to the USB_ENDPTCTRL*n* register can ensure both stall bits are set at the same instant.

NOTE

Any write to the USB_ENDPTCTRL*n* register during operational mode must preserve the endpoint type field (i.e., perform a read-modify-write).

Table 32-91. Device Controller Stall Response Matrix

USB Packet	Endpoint Stall Bit	Effect on STALL bit	USB Response
SETUP packet received by a non-control endpoint.	N/A	None	STALL
IN/OUT/PING packet received by a non-control endpoint.	1	None	STALL
IN/OUT/PING packet received by a non-control endpoint.	0	None	ACK/NAK/NYET
SETUP packet received by a control endpoint.	N/A	Cleared	ACK
IN/OUT/PING packet received by a control endpoint	1	None	STALL
IN/OUT/PING packet received by a control endpoint.	0	None	ACK/NAK/NYET

32.8.4.3 Data Toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe. For more information on data toggle, refer to the USB 2.0 specification.

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a '1' to the data toggle reset bit in the USB_ENDPTCTRL*n* register. This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

Data Toggle Inhibit is for test purposes only and should never be used during normal device controller operation.

Setting the data toggle inhibit bit active (1) causes the USB controller to ignore the data toggle pattern normally sent and accept all incoming data packets regardless of the data toggle state.

In normal operation, the USB controller checks the DATA0/DATA1 bit against the data toggle to determine if the packet is valid. If data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the USB controller from re-sending the same packet, the device controller responds to the error packet by acknowledging it with either an ACK or NYET response.

32.8.5 Device Operational Model For Packet Transfers

All transactions on the USB bus are initiated by the host and the device must respond to any request from the host within the turnaround time stated in the USB 2.0 Specification.

A USB host sends requests to the device controller (USB controller) in an order that cannot be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is

considered. For example, if endpoint 3 (transmit direction) is configured as a bulk pipe, you can expect the host to send IN requests to that endpoint. The device controller prepares packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as priming the endpoint. This term appears throughout the following documentation to describe the device controller (USB controller) operation so the DCD can be architected properly use priming. Further, the term flushing describes the action of clearing a packet queued for execution.

Priming a transmit endpoint causes the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH). After the dTD is fetched, it is stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to manage a request from the host without the need to follow the linked list, starting at the dQH when the host request is received.

After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller.

After a priming request is complete, an endpoint state of primed is indicated in the USB_ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB.

Because only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the transaction starts. The FIFO must be sized to account for the maximum latency that can be incurred by the system memory bus.

Priming receive endpoints are identical to priming of transmit endpoints from the point of view of the DCD. At the device controller, the major difference in the operational model is no data movement of the leading packet data simply because the data is to be received from the host.

As part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Therefore, the size of the RX FIFO does not scale with the number of endpoints.

32.8.5.1 Interrupt/Bulk Endpoint Operational Model

The behaviors of the device controller for interrupt and bulk endpoints are identical. All valid IN and OUT transactions to bulk pipes handshake with a NAK unless the endpoint had been primed. After the endpoint has been primed, data delivery commences.

A dTD is retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes n packets to be transferred according to the USB variable length transfer protocol. The formula and table on the following page describes how the device controller computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0

N = INT(Number Of Bytes/Max. Packet Length) + 1

With Zero Length Termination (ZLT) = 1

N = MAXINT(Number Of Bytes/Max. Packet Length)

Table 32-92. Variable Length Transfer Protocol Example (ZLT = 0)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	3	256	256	0
512	512	2	512	0	

Table 32-93. Variable Length Transfer Protocol Example (ZLT = 1)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	2	256	256	
512	512	1	512		

NOTE

The MULT field in the dQH must be set to 00 for bulk, interrupt, and control endpoints.

The ZLT bit in the dTD operates as following on BULK and control transfers:

ZLT = 0, the default value, means that the zero length termination is active. With the ZLT option enabled, when the device is transmitting, the hardware automatically appends a zero packet length when the following conditions are true:

- The packet transmitted equals maximum packet length.
- The dTD has exhausted the field Total Bytes

After this the dTD is retired. When the device is receiving, if the last packet length received equal maximum packet length and the total bytes is zero, it waits for a zero length packet from the host to retire the current dTD.

ZLT = 1, means the zero length termination is inactive. With the ZLT option disabled, when the device is transmitting, the hardware does not append any zero length packet. When receiving, it does not require a zero length packet to retire a dTD whose last packet was equal to the maximum packet length packet.

The dTD is retired as soon as Total Bytes field goes to zero, or a short packet is received.

Each transfer is defined by one dTD, so the zero length termination is for each dTD.

In some software application cases, the logic transfer does not fit into one dTD, so it does not make sense to add a Zero Length Termination packet each time a dTD is consumed. On those cases, it is recommended to turn off this ZLT feature and use software to generate the zero length termination.

TX-dTD is complete when:

- All packets described dTD were successfully transmitted. Total bytes in dTD equal zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. Total bytes in dTD equal zero when this occurs.
- A short packet (number of bytes < maximum packet length) was received. This is a successful transfer completion; DCD must check total bytes in dTD to determine the number of bytes that are remaining. From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.
- A long packet was received (number of bytes > maximum packet size) or (total bytes received > total bytes specified). This is an error condition. The device controller discards the remaining packet, and set the buffer error bit in the dTD. In addition, the endpoint is flushed and the USBERR interrupt becomes active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD is cleared and the next pointer is followed when the terminate bit is clear. When the terminate bit is set, the USB controller flushes the endpoint/direction and ceases operations for that endpoint/direction.

On the unsuccessful completion of a packet (see long packet above), the dQH is left pointing to the dTD that was in error. To recover from this error condition, the DCD must properly re-initialize the dQH by clearing the active bit and update the nextTD pointer before attempting to re-prime the endpoint.

NOTE

All packet level errors, such as a missing handshake or CRC error, are retried automatically by the device controller.

There is no required interaction with the DCD for managing such errors.

Table 32-94. Interrupt/Bulk Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	Ignore	Ignore	Ignore	N/A	N/A
In	STALL	NAK	Transmit	BS Error[1]	N/A
Out	STALL	NAK	Receive + NYET/ACK[2]	N/A	NAK
Ping	STALL	NAK	ACK	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

1 Force Bit Stuff Error

2 NYET/ACK — NYET unless the transfer descriptor has packets remaining according to the USB variable length protocol then ACK.

NOTE

System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

32.8.5.2 Control Endpoint Operation Model

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The USB controller always accepts the setup phase unless the setup lockout is engaged.

The setup lockout engages so future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, data is not written as it is being read potentially causing an invalid setup packet.

The setup lockout mechanism can be disabled and a tripwire type semaphore ensures the setup packet payload is extracted from the queue head without being corrupted by an incoming setup packet. This is preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

Setup Packet Handling:

- Disable setup lockout by writing 1 to setup lockout mode (SLOM) in the USB_USBMODE register (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

NOTE

Leaving the setup lockout mode as 0 results in a potential compliance issue.

- After receiving an interrupt and inspecting the USB_ENDPTSETUPSTAT register to determine that a setup packet was received on a particular pipe:
  - Write 1 to clear corresponding bit in the USB_ENDPTSETUPSTAT register.
  - Write 1 to setup TripWire (SUTW) in the USB_USBCMD register.
  - Duplicate contents of dQH.SetupBuffer into local software byte array.
  - Read setup TripWire (SUTW) in USB_USBCMD register. (if set - continue; if cleared - goto 2)
  - Write 0 to clear setup Tripwire (SUTW) in USB_USBCMD register.
  - Process setup packet using local software byte array copy and execute status/handshake phases.

NOTE

After receiving a new setup packet, the status and/or handshake phases may be pending from a previous control sequence. These should be flushed and de-allocated before linking a new status and/or handshake dTD for the most recent setup packet.

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by reading the USB_ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime completes when the associated bit in the USB_ENDPTPRIME register is 0 and the associated bit in the USB_ENDPTSTATUS register is 1. If a prime fails, i.e., the USB_ENDPTPRIME bit goes to zero and the USB_ENDPTSTATUS bit is not set, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the

USB_ENDPTPRIME bit is cleared, the transfer descriptor can be freed and the DCD must reinterpret the setup packet.

Should a setup arrive after the data stage is primed, the device controller automatically clears the USB_ENDPTSTATUS register to enforce data coherency with the setup packet.

NOTE

The MULT field in the dQH must be set to 00 for bulk, interrupt, and control endpoints.

NOTE

Error handling of data phase packets is the same as bulk packets described previously.

Similar to the data phase, DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase. The DCD must also perform the same checks of the USB_ENDPTSETUPSTAT as described above in the data phase.

NOTE

The MULT field in the dQH must be set to 00 for bulk, interrupt, and control endpoints.

NOTE

Error handling of data phase packets is the same as bulk packets described previously.

Table 32-95 shows the device controller response to packets on a control endpoint according to the device controller state.

Table 32-95. Control Endpoint Bus Response Matrix

Token Type	Endpoint State						Setup Lockout
	Stall	Not Primed	Primed	Underflow	Overflow	Not Enabled	
Setup	ACK	ACK	ACK	N/A	SYSERR[1]	BTO	—
In	STALL	NAK	Transmit	BS Error[2]	N/A	BTO	N/A
Out	STALL	NAK	Receive + NYET/ACK[3]	N/A	NAK	BTO	N/A
Ping	STALL	NAK	ACK	N/A	N/A	BTO	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	BTO	Ignore

1 SYSERR — System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

2 Force Bit Stuff Error

3 NYET/ACK — NYET unless the transfer descriptor has packets remaining according to the USB variable length protocol then ACK.

32.8.5.3 Isochronous Endpoint Operational Model

Isochronous endpoints are used for real-time scheduled delivery of data and their operational model is significantly different than the host throttled bulk, interrupt, and control data pipes. Real time delivery by the USB controller is accomplished by:

- Exactly MULT Packets per (micro) Frame are transmitted/received.

NOTE

MULT is a two-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.

- NAK responses are not used. Instead, zero length packets are sent in response to an IN request to an unprimed endpoint. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro)frame. If the ISO-dTD remains active after that frame, the ISO-dTD is held ready until executed or canceled by the DCD.

The USB controller in host mode uses the periodic frame list to schedule data exchanges to isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for control/bulk/interrupt endpoints is also for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is priming an ISO-endpoint is a delayed operation such that an endpoint becomes primed only after a SOF is received. After the DCD writes the prime bit, the prime bit is cleared as usual to indicate to software the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD, but occurs so the device controller can match the dTD to a specific (micro)frame.

Another difference with isochronous endpoints is the transaction must wholly complete in a (micro)frame. After an ISO transaction is started in a (micro)frame, it retires the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition.

The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller forces retirement of the ISO-dTD and move to the next ISO-dTD.

Fulfillment errors are caused only by partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction stays primed indefinitely. Thus, software must discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

The last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the transaction error bit and the data is stored as usual for the application software to sort out.

- TX Packet Retired
  - MULT counter reaches zero.
  - Fulfillment Error [Transaction Error bit is set]

— #Packets Occurred > 0 AND # Packets Occurred < MULT

NOTE

For TX-ISO, MULT counter can be loaded with a lesser value in the dTD multiplier override field. If the multiplier override is zero, the MULT counter is initialized to the Multiplier in the QH.

- RX Packet Retired:
  - — MULT counter reaches zero.
  - — Non-MDATA Data PID is received
  - — Overflow Error:
  - — Packet received is > maximum packet length. [Buffer Error bit is set]
  - — Packet received exceeds total bytes allocated in dTD. [Buffer Error bit is set]
  - — Fulfillment Error [Transaction Error bit is set]
  - — # Packets Occurred > 0 AND # Packets Occurred < MULT
  - — CRC Error [Transaction Error bit is set]

NOTE

For ISO, when a dTD is retired, the next dTD is primed for the next frame. For continuous (micro)frame-to-(micro)frame operation, DCD should ensure the dTD linked-list is out ahead of the device controller by at least two (micro)frames.

32.8.5.3.1 Isochronous Pipe Synchronization

When necessary to synchronize an isochronous data pipe to the host, the (micro)frame number (USB_FRINDEX register) can act as a marker. To cause a packet transfer to occur at a specific (micro)frame number [N], DCD should interrupt on SOF during frame N-1. When USB_FRINDEX = N – 1, the DCD must write the prime bit. The USB controller primes the isochronous endpoint in (micro)frame N-1 so that the device controller executes delivery during (micro)frame N.

CAUTION

Priming an endpoint towards the end of (micro)frame N-1 does not guarantee delivery in (micro)frame N. The delivery may actually occur in (micro)frame N+1 if device controller does not have enough time to complete the prime before the SOF for packet N is received.

Table 32-96. Isochronous Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	STALL	STALL	STALL	N/A	N/A
In	NULL[1] Packet	NULL Packet	Transmit	BS Error[2]	N/A
Out	Ignore	Ignore	Receive	N/A	Drop Packet

Table 32-96. Isochronous Endpoint Bus Response Matrix (continued)

	Stall	Not Primed	Primed	Underflow	Overflow
Ping	Ignore	Ignore	Ignore	Ignore	Ignore
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

1 Zero Length Packet
2 Force Bit Stuff Error

32.8.6 Managing Queue Heads

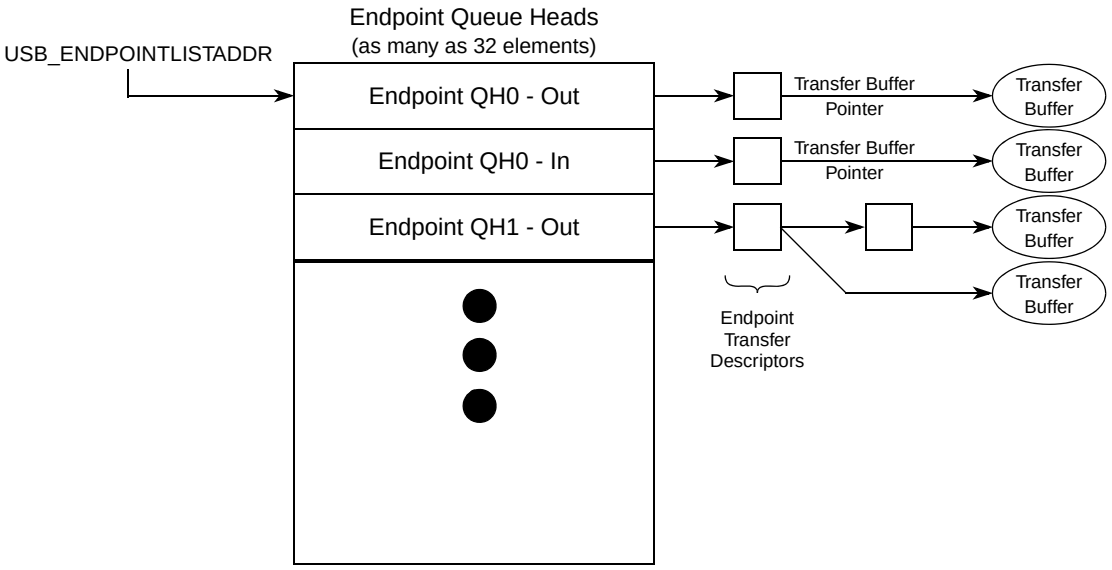


Figure 32-75. Endpoint Queue Head Diagram

The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device transfer descriptor (dTD). An area of memory pointed to by USB_ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown in Figure 32-75. The even elements in the list of dQH's are receive endpoints (OUT/SETUP) and the odd elements are for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. After the dTD has been retired, it is no longer part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors because pointers no longer exist within the queue head once the dTD is retired (see section Software Link Pointers).

In addition to the current and next pointers and the dTD overlay examined in the section on operational model for packet transfers, dQH also contains the following parameters for the associated endpoint: multiplier, maximum packet length, interrupt on setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

32.8.6.1 Queue Head Initialization

One pair of device queue heads must be initialized for each active endpoint. To initialize a device queue head:

- Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1,2, or 3 as required bandwidth with the USB Chapter 9 protocol.

NOTE

In FS mode, the multiplier field can only be 1 for ISO endpoints.

- Write the next dTD Terminate bit field to 1.
- Write the active bit in the status field to 0.
- Write the halt bit in the status field to 0.

NOTE

The DCD must only modify dQH if the associated endpoint is not primed and there are no outstanding dTD's.

32.8.6.2 Operational Model For Setup Transfers

As discussed in Section 32.8.5.2, “Control Endpoint Operation Model,” setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD, but stores the incoming data from a setup packet in an 8-byte buffer within the dQH instead.

Upon receiving notification of the setup packet, DCD should manage setup transfer as demonstrated here:

1. Copy setup buffer contents from dQH - RX to software buffer.
2. Acknowledge setup backup by writing a 1 to the corresponding bit in the USB_ENDPTSETUPSTAT register.

NOTE

The acknowledge must occur before continuing to process the setup packet.

NOTE

After the acknowledge has occurred, the DCD must not attempt to access the setup buffer in the dQH - RX. Only the local software copy should be examined.

3. Check for pending data or status dTDs from previous control transfers and flush if any exist as discussed in section flushing/de-priming an endpoint.

NOTE

It is possible for the device controller to receive setup packets before previous control transfers complete. Existing control packets in progress must be flushed and the new control packet completed.

4. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

32.8.7 Managing Transfers with Transfer Descriptors

32.8.7.0.1 Software Link Pointers

It is necessary for the DCD software to maintain head and tail pointers for the linked list of dTDs for each respective queue head. This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in the next section for managing dTD assumes the DCD can use reference for the head and tail of the dTD linked list.

NOTE

To conserve memory, reserved fields at the end of the dQH can store the head and tail pointers, but it remains the responsibility of the DCD to maintain the pointers.

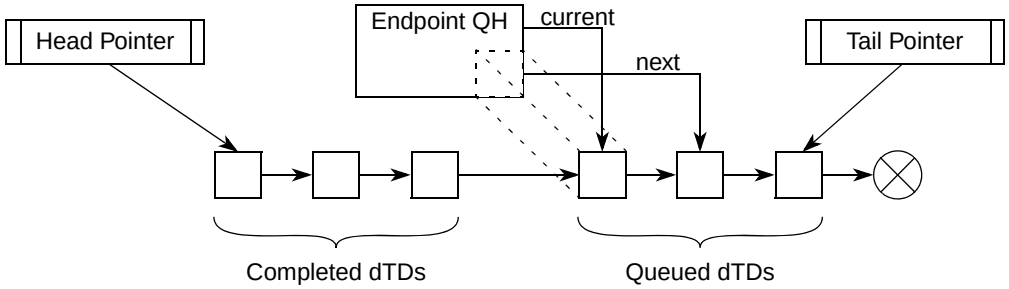


Figure 32-76. Software Link Pointers

32.8.7.1 Building a Transfer Descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer. Use the following procedure for building dTDs.

Allocate 8 32-bit word dTD block of memory aligned to 8 32-bit word boundaries. Example: bit address 4:0 would be equal to 00000.

Write the following fields:

1. Initialize first 7 32-bit words to 0.
2. Set the terminate bit to 1.
3. Fill in total bytes with transfer size.
4. Set the interrupt on complete if desired.
5. Initialize the status field with the active bit set to 1 and all remaining status bits set to 0.
6. Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
7. Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

32.8.7.2 Executing A Transfer Descriptor

To safely add a dTD, DCD must follow this procedure which manages the event where the device controller reaches the end of the dTD list at the same time a new dTD is added to the end of the list.

Determine whether the link list is empty:

Check DCD driver to see if pipe is empty (internal representation of linked-list should indicate if any packets are outstanding)

Case 1: Link list is empty

8. Write dQH next pointer AND dQH terminate bit to 0 as a single 32-bit word operation.
9. Clear active and halt bit in dQH (in case set from a previous error).
10. Prime endpoint by writing 1 to correct bit position in the USB_ENDPTPRIME register.

Case 2: Link list is not empty

1. Add dTD to end of linked list.
2. Read correct prime bit in the USB_ENDPTPRIME register - if 1 DONE.
3. Set ATDTW bit in USB_USBCMD register to 1.
4. Read correct status bit in USB_ENDPTSTATUS. (store in temporary variable for later)
5. Read ATDTW bit in the USB_USBCMD register.

   If 0, goto 3.

   If 1, continue to 6.
6. Write ATDTW bit in USB_USBCMD register to '0'.
7. If status bit read in (3) is '1' DONE.
8. If status bit read in (3) is '0' then Goto Case 1: Step 1.

32.8.7.3 Transfer Completion

After a dTD has been initialized and the associated endpoint primed, the device controller executes the transfer upon the host-initiated request. The DCD is notified with a USB interrupt if the interrupt on complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

CAUTION

Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs have finished (active bit cleared).

By reading the status fields of the completed dTDs, DCD can determine if the transfers completed successfully. Success is determined with this combination of status bits:

- Active = 0
- Halted = 0
- Transaction Error = 0

- Data Buffer Error = 0

Should any combination other than the one shown above exist, DCD must take proper action. Transfer failure mechanisms are indicated in the device error matrix.

In addition to checking the status bit, DCD must read the transfer bytes field to determine the actual bytes transferred. When a transfer is complete, the total bytes transferred is decremented by the actual bytes transferred. For transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according to the USB variable length packet protocol.

32.8.7.4 Flushing/De-priming an Endpoint

It is necessary for the DCD to flush to de-prime one more endpoints on a USB device reset or during a broken control transfer. There may also be application specific requirements to stop transfers in progress. The DCD can use the following procedure to stop a transfer in progress:

1. Write a 1 to the corresponding bit(s) in USB_ENDPTFLUSH.
2. Wait until all bits in USB_ENDPTFLUSH are 0.

NOTE

This operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.

3. Read USB_ENDPTSTATUS to ensure all endpoints commanded to be flushed are now 0. If the corresponding bits are 1 after the second step has finished, flush has failed as described here:

   In rare cases, a packet is in progress to the particular endpoint is commanded to flush using USB_ENDPTFLUSH. A safeguard is in place to refuse the flush to ensure the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush by repeating steps one through three until each endpoint is successfully flushed.

32.8.8 Device Error Matrix

The following table summarizes packet errors that are not automatically managed by the USB controller.

Table 32-97. Device Error Matrix

Error	Direction	Packet Type	Data Buffer Error Bit	Transaction Error Bit
Overflow[1]	RX	Any	1	0
ISO Packet Error	RX	ISO	0	1
ISO Fulfillment Error	Both	ISO	0	1

[1] This error also set the halt bit in the dQH and if there are dTDs remaining in the linked list for the endpoint, those are not executed.

The device controller manages all errors on bulk/control/interrupt endpoints except for a data buffer overflow. However, for ISO endpoints, errors packets are not retried and errors are tagged as indicated.

Table 32-98. Error Descriptions

Overflow	Number of bytes received exceeded max. packet size or total buffer length.
ISO Packet Error	CRC Error on received ISO packet. Contents not guaranteed to be correct.
ISO Fulfillment Error	Host failed to complete the number of packets defined in the dQH mult field within the given (micro)frame. For scheduled data delivery, DCD may need to readjust the data queue because a fulfillment error causes device controller to cease data transfers on the pipe for one (micro)frame. During dead (micro)frame, device controller reports error on the pipe and primes for the next frame.

32.8.9 Servicing Interrupts

The interrupt service routine must consider there are high-frequency, low-frequency, and error operations to order accordingly.

32.8.9.1 High-Frequency Interrupts

High frequency interrupts in particular should be managed in the order below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

Table 32-99. Interrupt Handling Order

Execution Order	Interrupt	Action
1a	USB Interrupt[1] ENDPTSETUPSTATUS	Copy contents of setup buffer and acknowledge setup packet (as indicated in section Managing Queue Heads). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol.
1b	USB Interrupt ENDPTCOMPLETE	Manages completion of dTD as indicated in section Managing Queue Heads.
2	SOF Interrupt	Action as deemed necessary by application. This interrupt may not have a use in all applications.

1 It is likely that multiple interrupts to stack up on any call to the interrupt service routine and during the interrupt service routine.

32.8.9.2 Low-Frequency Interrupts

The low-frequency events include the following interrupts. These interrupts can be managed in any order because they don't occur often in comparison to the high-frequency interrupts.

Table 32-100. Low Frequency Interrupt Events

Interrupt	Action
Port Change	Change software state information.
Sleep Enable (Suspend)	Change software state information. Low power handling as necessary.
Reset Received	Change software state information. Abort pending transfers.

32.8.9.3 Error Interrupts

Error interrupts are least frequent and should be placed last in the interrupt service routine.

Table 32-101. Error Interrupt Events

Interrupt	Action
USB Error Interrupt.	This error is redundant because it combines USB Interrupt and an error status in the dTD. The DCD more aptly manages packet-level errors by checking dTD status field upon receipt of USB Interrupt (w/ ENDPTCOMPLETE).
System Error	Unrecoverable error. Immediate Reset of core; free transfers buffers in progress and restart the DCD.

32.8.10 Deviations from the EHCI Specifications

The host mode operation of the modules is nearly EHCI-compatible with few minor differences. For the most part, the modules conform to the data structures and operations described in Section 3, “Data Structures,” and Section 4, “Operational Model,” in the EHCI specification. The particulars of the deviations occur in these areas:

- Device operation (OTG module only)—In host mode, the device operational registers are generally disabled and device mode is mostly transparent when in host mode. However, there are exceptions documented in the following sections.
- Embedded design interface—The modules do not have a PCI interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.

For the purposes of the USB controller implementing dual-role host/device controller with support for OTG applications, it is necessary to deviate from the EHCI specification. Device operation and OTG operation are not specified in the EHCI, and implementation supported in the OTG module is proprietary.

32.8.10.1 Device Operation

The co-existence of a device operational controller within the OTG module has little effect on EHCI compatibility for host operation. However, because the OTG controller is initialized in neither host nor device mode, the USB_USBMODE register must be programmed for host operation before the EHCI host controller driver can begin EHCI host operations.

32.8.10.2 Non-Zero Fields in the Register File

Some of the reserved fields and reserved addresses in the capability registers and operational registers have meaning in device mode; therefore, the following must be adhered to:

- Write operations to all EHCI reserved fields (some of which are device fields in the OTG module) in the operation registers should always be written to zero. This is an EHCI requirement of the device controller driver that must be adhered to.
- Read operations by the module must properly mask EHCI reserved fields (some of which are device fields in the OTG module registers).

32.8.10.3 SOF Interrupt

The SOF interrupt is a free running 125 µsec interrupt for host mode. EHCI does not specify this interrupt, but it has been added for convenience and as a potential software time base. The free running interrupt is shared with the OTG-only device-mode start-of-frame interrupt. See Section 32.2.4.2, “USB Status

(USB_USBSTS) Register," and Section 32.2.4.3, "USB Interrupt Enable Register (USB_USBINTR)," for more information.

32.8.10.4 Embedded Design

This is an embedded USB host controller as defined by the EHCI specification and does not implement the PCI configuration registers.

32.8.10.4.1 Frame Adjust Register

Because the optional PCI configuration registers are not included in this implementation, there is no corresponding bit level timing adjustments like those provided by the frame adjust register in the PCI configuration registers. Starts of microframes are timed precisely to 125 µsec using the transceiver clock as a reference clock.

32.8.10.5 Miscellaneous Variations from EHCI

32.8.10.5.1 Programmable Physical Interface Behavior

The modules support multiple physical interfaces which can operate in different modes when the module is configured with the software programmable physical interface modes. The control bits for selecting the PHY operating mode have been added to the USB_PORTSC*n* register providing a capability defined by EHCI specification.

32.8.10.5.2 Discovery

The port connect methods specified by EHCI require setting the port reset bit in the register for a duration of 10 msec. Due to the complexity required to support the attachment of devices that are not high speed, counters are already present in the design that can count the 10 msec reset pulse to alleviate the requirement of the software to measure this duration. Therefore, the basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a 1 to the reset bit of the device.
- Software shall write a 0 to the reset bit of the device after 10 msec.
  - This step (necessary in a standard EHCI design) may be omitted with this implementation. Should the EHCI host controller driver attempt to write a 0 to the reset bit while a reset is in progress, the write is ignored and the reset continues until completion.
- [Port Change Interrupt] Port enable change occurs to notify the host controller the device is now operational, and at this point, the port speed has been determined.

After the port change interrupt indicates a port is enabled, EHCI stack should determine the port speed. Unlike EHCI implementation, which re-assigns the port owner for any device that does not connect at high-speed, this host controller supports direct attach of non-HS devices. Therefore, the following differences are important regarding port speed detection:

- Port owner is read-only and always reads 0.

- A 2-bit port speed indicator has been added to USB_PORTSC*n* to provide the current operating speed of the port to the host controller driver.
- A 1-bit high speed indicator has been added to USB_PORTSC*n* to signify that the port is in HS vs. FS/LS – This information is redundant with the 2-bit port speed indicator above.

This page is intentionally left blank.

Appendix A
Memory Map

A.1 Introduction

This appendix lists the memory-mapped registers and addresses for the MPC5125 device.

A.2 MPC5125 Register Map

Table A-1. MPC5125 memory map

Offset from IMMRBAR (0xFF40_0000)	Absolute offset[1]	Region Name	Section/Page
0x0_0000–0_01FF	0xFF40_0000	System configuration (XLBMEN)	Appendix A/A-1[2]
0x0_0200–0_08FF	0xFF40_0200	Reserved[3]	
0x0_0900–0_09FF	0xFF40_0900	Software watchdog timer (WDT)	Chapter 29/29-1
0x0_0A00–0_0AFF	0xFF40_0A00	Real time clock (RTC)	Chapter 27/27-1
0x0_0B00–0_0BFF	0xFF40_0B00	General purpose timer (GPT1)	Chapter 15/15-1
0x0_0C00–0_0CFF	0xFF40_0C00	Integrated programmable interrupt controller (IPIC)	Chapter 18/18-1
0x0_0D00–0_0DFF	0xFF40_0D00	CSB arbiter	Chapter 8/8-1
0x0_0E00–0_0EFF	0xFF40_0E00	Reset module (RESET)	Chapter 4/4-1
0x0_0F00–0_0FFF	0xFF40_0F00	Clock module (CLOCK)	Chapter 5/5-1
0x0_1000–0_10FF	0xFF40_1000	Power management control (PMC)	Chapter 24/24-1
0x0_1100–0_117F	0xFF40_1100	General Purpose I/O (GPIO1)	Chapter 16/16-1
0x0_1180–0_11FF	0xFF40_1180	General Purpose I/O (GPIO2)	Chapter 16/16-1
0x0_1200–0_12FF	0xFF40_1200	Reserved	
0x0_1300–0_137F	0xFF40_1300	MSCAN1	Chapter 22/22-1
0x0_1380–0_13FF	0xFF40_1380	MSCAN2	Chapter 22/22-1
0x0_1400–0_14FF	0xFF40_1400	Byte data link controller (BDLC)	Chapter 6/6-1
0x0_1500–0_15FF	0xFF40_1500	Secure digital host controller (SDHC1)	Chapter 28/28-1
0x0_1600–0_16FF	0xFF40_1600	Reserved	
0x0_1700–0_171F	0xFF40_1700	Inter-integrated circuit (I^2C1)	Chapter 19/19-1
0x0_1720–0_173F	0xFF40_1720	Inter-integrated circuit (I^2C2)	Chapter 19/19-1
0x0_1740–0_17FF	0xFF40_1740	Inter-integrated circuit (I^2C3)	Chapter 19/19-1

Table A-1. MPC5125 memory map (continued)

Offset from IMMRBAR (0xFF40_0000)	Absolute offset[1]	Region Name	Section/Page
0x0_1800–0_1FFF	0xFF40_1800	Reserved	
0x0_2000–0_20FF	0xFF40_2000	Reserved	
0x0_2100–0_21FF	0xFF40_2100	Display Interface Unit (DIU)	Chapter 10/10-1
0x0_2200–0_22FF	0xFF40_2200	Reserved	
0x0_2300–0_237F	0xFF40_2300	MSCAN3	Chapter 22/22-1
0x0_2380–0_23FF	0xFF40_2380	MSCAN4	Chapter 22/22-1
0x0_2400–0_27FF	0xFF40_2400	Reserved	
0x0_2800–0_2FFF	0xFF40_2800	Fast Ethernet Controller (FEC1)	Chapter 14/14-1
0x0_3000–0_33FF	0xFF40_3000	USB ULPI1	Chapter 32/32-1
0x0_3400–0_3FFF	0xFF40_3400	Reserved	
0x0_4000–0_43FF	0xFF40_4000	USB ULPI2	Chapter 32/32-1
0x0_4400–0_47FF	0xFF40_4400	Reserved	
0x0_4800–0_4FFF	0xFF40_4800	Fast Ethernet Controller (FEC2)	Chapter 14/14-1
0x0_5000–0_50FF	0xFF40_5000	General purpose timer (GPT2)	Chapter 15/15-1
0x0_5100–0_51FF	0xFF40_5100	Secure digital host controller (SDHC2)	Chapter 28/28-1
0x0_5200–0_8FFF	0xFF40_5200	Reserved	
0x0_9000–0_9FFF	0xFF40_9000	Multi-port DRAM controller (MDDRC)	Chapter 11/11-1
0x0_A000–0_AFFF	0xFF40_A000	I/O control	Chapter 20/20-1
0x0_B000–0_BFFF	0xFF40_B000	IC identification module (IIM)	Chapter 17/17-1
0x0_C000–0_FFFF	0xFF40_C000	Reserved	
0x1_0000–1_01FF	0xFF41_0100	LocalPlus controller (LPC)	Chapter 21/21-1
0x1_0200–1_0FFF	0xFF41_02--	Reserved	
0x1_1000–1_10FF	0xFF41_1000	Programmable Serial Controller 0 (PSC0)	Chapter 25/25-1
0x1_1100–1_11FF	0xFF41_1100	Programmable Serial Controller 1 (PSC1)	Chapter 25/25-1
0x1_1200–1_12FF	0xFF41_1200	Programmable Serial Controller 2 (PSC2)	Chapter 25/25-1
0x1_1300–1_13FF	0xFF41_1300	Programmable Serial Controller 3 (PSC3)	Chapter 25/25-1
0x1_1400–1_14FF	0xFF41_1400	Programmable Serial Controller 4 (PSC4)	Chapter 25/25-1
0x1_1500–1_15FF	0xFF41_1500	Programmable Serial Controller 5 (PSC5)	Chapter 25/25-1
0x1_1600–1_16FF	0xFF41_1600	Programmable Serial Controller 6 (PSC6)	Chapter 25/25-1
0x1_1700–1_17FF	0xFF41_1700	Programmable Serial Controller 7 (PSC7)	Chapter 25/25-1
0x1_1800–1_18FF	0xFF41_1800	Programmable Serial Controller 8 (PSC8)	Chapter 25/25-1

Table A-1. MPC5125 memory map (continued)

Offset from IMMRBAR (0xFF40_0000)	Absolute offset[1]	Region Name	Section/Page
0x1_1900–1_19FF	0xFF41_1900	Programmable Serial Controller 9 (PSC9)	Chapter 25/25-1
0x1_1A00–1_1EFF	0xFF41_1A00	Reserved	
0x1_1F00–1_1FFF	0xFF41_1F00	Serial FIFO (SFIFO) for PSC 0 to 9	Chapter 25/25-1
0x1_2000–1_3FFF	0xFF41_2000	Reserved	
0x1_4000–1_57FF	0xFF41_4000	DMA	Chapter 9/9-1
0x1_5800–F_FFFF	0xFF41_5800	Reserved	

1 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, "System Configuration and Memory Map (XLBMEN + Mem Map)."

2 This table provides the memory map.

3 Reserved locations are not guaranteed to be empty. Software should not access reserved locations.

Table A-2. MPC5125 Detailed Register Map

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
System Configuration (XLBMEN) 0xFF40_0000[2] **Chapter 2, "System Configuration and Memory Map (XLBMEN + Mem Map)"**				
0x000	Internal Memory Map Base Address Register (IMMRBAR)	R/W	0xFF80_0000	2.2.5.1.1/2-5
0x004–0x01F	Reserved			
0x020	LocalPlus Boot Access Window register (LPBAW)	R/W	See register description.	2.2.5.1.3/2-8
0x024	LocalPlus CS0 Access Window register (LPCS0AW)	R/W	See register description.	2.2.5.1.4/2-9
0x028	LocalPlus CS1 Access Window register (LPCS1AW)	R/W	See register description.	2.2.5.1.4/2-9
0x02C	LocalPlus CS2 Access Window register (LPCS2AW)	R/W	See register description.	2.2.5.1.4/2-9
0x030	LocalPlus CS3 Access Window register (LPCS3AW)	R/W	See register description.	2.2.5.1.4/2-9
0x034	LocalPlus CS4 Access Window register (LPCS4AW)	R/W	See register description.	2.2.5.1.4/2-9
0x038	LocalPlus CS5 Access Window register (LPCS5AW)	R/W	See register description.	2.2.5.1.4/2-9
0x03C	LocalPlus CS6 Access Window register (LPCS6AW)	R/W	See register description.	2.2.5.1.4/2-9

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x040	LocalPlus CS7 Access Window register (LPCS7AW)	R/W	See register description.	2.2.5.1.4/2-9
0x044–0x09F	Reserved			
0x0A0	DDR Local Access Window0 Base Address register (DDRLAWBAR0)	R/W	0x1000_0000	2.2.5.1.5/2-10
0x0A4	DDR Local Access Window0 Attribute register (DDRLAWAR0)	R/W	0x0000_0019	2.2.5.1.6/2-11
0x0A8–0x0C3	Reserved			
0x0C4	SRAM Address Register (SRAMBAR)	R/W	0x3000_0000	2.2.5.1.7/2-11
0x0C8	NFC Address Register (NFCBAR)	R/W	See register description.	2.2.5.1.8/2-12
0x0CC–0x0FF	Reserved			
Software Watchdog Timer (WDT) 0xFF40_0900 **Chapter 29, "Software Watchdog Timer (WDT)"**				
0x00	Reserved			
0x04	WDT_SWCRR—Software watchdog control register	R/W	0x0000_0000	29.2.2.1/29-2
0x08	WDT_SWCNR—Software watchdog count register	R	0x0000_0000	29.2.2.2/29-3
0x0C–0x0D	Reserved			
0x0E	SWSRR—Software watchdog service register	W	0x0004	29.2.2.3/29-4
0x0004–0xFF	Reserved			
Real Time Clock (RTC) 0xFF40_0A00[2] **Chapter 27, "Real Time Clock (RTC)"**				
0x00	RTC_TSR—RTC Time Set Register	R/W	0x0000_0000	27.3.2.1/27-4
0x04	RTC_DSR—RTC Date Set Register	R/W	0x0000_0000	27.3.2.2/27-6
0x08	RTC_NY_STP—RTC New Year and Stopwatch Register	R/W	0x0000_0000	27.3.2.3/27-7
0x0C	RTC_ALM_IE—RTC Alarm and Interrupt Enable Register	R/W	0x0000_0008	27.3.2.4/27-8
0x10	RTC_CTR—RTC Current Time Register	R	0x0000_0000	27.3.2.5/27-9
0x14	RTC_CDR—RTC Current Date Register	R	0x0000_0000	27.3.2.6/27-9
0x18	RTC_ALM_STP_INT—RTC Alarm and Stopwatch Interrupt Register	R/W	0x0000_0000	27.3.2.7/27-10
0x1C	RTC_PI_BE—RTC Periodic Interrupt and Bus Error Register	R/W	0x0000_0000	27.3.2.8/27-11
0x20	RTC_TTR—RTC Target Time Register	R/W	0x0000_0000	27.3.2.9/27-11
0x24	RTC_ATC—RTC Actual Time Counter Register	R	0xFFFF_FFFE	27.3.2.10/27-12

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x28	RTC_KAR—RTC Keep Alive Register	R/W	0x0000_0087	27.3.2.11/27-13
0x2C–0xFF	Reserved			
General Purpose Timer (GPT1) 0xFF40_0B00 **Chapter 15, "General Purpose Timers (GPT)"**				
0x00	GPT0 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x04	GPT0 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x08	GPT0 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x0C	GPT0 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x10	GPT1 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x14	GPT1 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x18	GPT1 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x1C	GPT1 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x20	GPT2 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x24	GPT2 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x28	GPT2 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x2C	GPT2 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x30	GPT3 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x34	GPT3 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x38	GPT3 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x3C	GPT3 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x40	GPT4 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x44	GPT4 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x48	GPT4 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x4C	GPT4 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x50	GPT5 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x54	GPT5 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x58	GPT5 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x5C	GPT5 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x60	GPT6 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x64	GPT6 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x68	GPT6 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x6C	GPT6 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x70	GPT7 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x74	GPT7 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x78	GPT7 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x7C	GPT7 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x80–0xFF	Reserved			
Integrated Programmable Interrupt Controller (IPIC) 0xFF40_0C00[2] **Chapter 18, "Integrated Programmable Interrupt Controller (IPIC)"**				
0x00	System Global Interrupt Configuration Register (IPIC_SICFR)	R/W	0x0000_0000	18.2.1.1/18-6
0x04	System Global Interrupt Vector Register (IPIC_SIVCR)	R	0x0000_0000	18.2.1.2/18-8
0x08	System Internal Interrupt Pending Register (IPIC_SIPNR_H)	R	0x0000_0000	18.2.1.3/18-11
0x0C	System Internal Interrupt Pending Register (IPIC_SIPNR_L)	R	0x0000_0000	18.2.1.3/18-11
0x10	System Internal Interrupt Group A Priority Register (IPIC_SIPRR_A)	R/W	0x0530_9770	18.2.1.4/18-14
0x14	System Internal Interrupt Group B Priority Register (IPIC_SIPRR_B)	R/W	0x0530_9770	18.2.1.5/18-15
0x18	System Internal Interrupt Group C Priority Register (IPIC_SIPRR_C)	R/W	0x0530_9770	18.2.1.6/18-16
0x1C	System Internal Interrupt Group D Priority Register (IPIC_SIPRR_D)	R/W	0x0530_9770	18.2.1.7/18-17
0x20	System Internal Interrupt Mask Register (IPIC_SIMSR_H)	R/W	0x0000_0000	18.2.1.8/18-18
0x24	System Internal Interrupt Mask Register (IPIC_SIMSR_L)	R/W	0x0000_0000	18.2.1.8/18-18
0x28	System Internal Interrupt Control Register (IPIC_SICNR)	R/W	0x0000_0000	18.2.1.9/18-21
0x2C	System External Interrupt Pending Register (IPIC_SEPNR)	R/W	0xU000_0000	18.2.1.10/18-22
0x30	System Mixed Interrupt Group A Priority Register (IPIC_SMPRR_A)	R/W	0x0530_9770	18.2.1.11/18-23
0x34	System Mixed Interrupt Group B Priority Register (IPIC_SMPRR_B)	R/W	0x0530_9770	18.2.1.12/18-24

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x38	System External Interrupt Mask Register (IPIC_SEMSR)	R/W	0x0000_0000	18.2.1.13/18-25
0x3C	System External Interrupt Control Register (IPIC_SECNR)	R/W	0x0000_0000	18.2.1.14/18-26
0x40	System Error Status Register (IPIC_SERSR)	R/W	0x0000_0000	18.2.1.15/18-27
0x44	System Error Mask Register (IPIC_SERMR)	R/W	0xE080_0000	18.2.1.16/18-28
0x48	Reserved			
0x4C	System External Interrupt Polarity Check Register (IPIC_SEPCR)	R/W	0x0000_0000	18.2.1.17/18-29
0x50	System Internal Interrupt Force Register (IPIC_SIFCR_H)	R/W	0x0000_0000	18.2.1.18/18-29
0x54	System Internal Interrupt Force Register (IPIC_SIFCR_L)	R/W	0x0000_0000	18.2.1.18/18-29
0x58	System External Interrupt Force Register (IPIC_SEFCR)	R/W	0x0000_0000	18.2.1.19/18-32
0x5C	System Error Force Register (IPIC_SERFR)	R/W	0x0000_0000	18.2.1.20/18-33
0x60	System Critical Interrupt Vector Register (IPIC_SCVCR)	R	0x0000_0000	18.2.1.21/18-33
0x64	SysteM Management Interrupt Vector Register (IPIC_SMVCR)	R	0x0000_0000	18.2.1.22/18-34
0x68–0xFF	Reserved			
CSB Arbiter 0xFF40_0D00[2] **Chapter 8, "CSB Arbiter and Bus Monitor"**				
0x00	ACR—Arbiter Configuration Register	R/W	0x0UU0_0000	8.2.1.1/8-2
0x04	ATR—Arbiter Timers Register	R/W	0xFFFF_FFFF	8.2.1.2/8-4
0x08	ATER—Arbiter Transfer Error Register	R/W	0x0000_003F	8.2.1.3/8-5
0x0C	AER—Arbiter Event Register	R/W	0x0000_0000	8.2.1.4/8-6
0x10	AIDR—Arbiter Interrupt Definition Register	R/W	0x0000_0000	8.2.1.5/8-7
0x14	AMR—Arbiter Mask Register	R/W	0x0000_0000	8.2.1.6/8-8
0x18	AEATR—Arbiter Event Attributes Register	R	0x0000_0000	8.2.1.7/8-9
0x1C	AEADR—Arbiter Event Address Register	R	0x0000_0000	8.2.1.8/8-11
0x20	AERR—Arbiter Event Response Register	R/W	0x0000_0000	8.2.1.9/8-12
0x24–0xFF	Reserved			
Reset Module (RESET) 0xFF40_0E00[2] **Chapter 4, "Reset"**				
0x00	Reset Configuration Word Low (RCWL) Register	R	0x0U0U_0U00	4.6.1/4-9
0x04	Reset Configuration Word High (RCWH) Register	R	0xUUU0_UU00	4.6.2/4-9
0x08–0x0F	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x10	Reset Status Register (RSR)	R/W	0x6000_0000	4.6.3/4-10
0x14	Reset Mode Register (RMR)	R/W	0x0000_0000	4.6.4/4-12
0x18	Reset Protection Register (RPR)	R/W	0x0000_0000	4.6.5/4-13
0x1C	Reset Control Register (RCR)	R/W	0x0000_0000	4.6.6/4-13
0x20	Reset Control Enable Register (RCER)	R/W	0x0000_0000	4.6.7/4-14
0x24–0xFF	Reserved			
Clock Module (CLOCK) 0xFF40_0F00[2] **Chapter 5, "Clocks and Low-Power Modes"**				
0x00	System PLL Mode Register (SPMR)	R	0x0U0U_0000	5.3.1.1/5-8
0x04	System Clock Control Register 1 (SCCR1)	R/W	0xE000_1C00	5.3.1.2/5-9
0x08	System Clock Control Register 2 (SCCR2)	R/W	0x2040_0000	5.3.1.3/5-11
0x0C	System Clock Frequency Register 1 (SCFR1)	R/W	0x0180_100C	5.3.1.4/5-12
0x10	System Clock Frequency Register 2 (SCFR2)	R/W	0xUU00_0808	5.3.1.5/5-14
0x14	System Clock Frequency Shadow Register 2 (SCFR2S)	R/W	0xUU0U_0000	5.3.1.6/5-15
0x18	Bread Crumb Register (BCR)	R/W	—[3]	5.3.1.7/5-16
0x1C	PSC0 Clock Control Register (P0CCR)	R/W	0xFFFE_0000	5.3.1.8/5-17
0x20	PSC1 Clock Control Register (P1CCR)	R/W	0xFFFE_0000	5.3.1.9/5-18
0x24	PSC2 Clock Control Register (P2CCR)	R/W	0xFFFE_0000	5.3.1.10/5-19
0x28	PSC3 Clock Control Register (P3CCR)	R/W	0xFFFE_0000	5.3.1.11/5-20
0x2C	PSC4 Clock Control Register (P4CCR)	R/W	0xFFFE_0000	5.3.1.12/5-21
0x30	PSC5 Clock Control Register (P5CCR)	R/W	0xFFFE_0000	5.3.1.13/5-21
0x34	PSC6 Clock Control Register (P6CCR)	R/W	0xFFFE_0000	5.3.1.14/5-22
0x38	PSC7 Clock Control Register (P7CCR)	R/W	0xFFFE_0000	5.3.1.15/5-23
0x3C	PSC8 Clock Control Register (P8CCR)	R/W	0xFFFE_0000	5.3.1.16/5-24
0x40	PSC9 Clock Control Register (P9CCR)	R/W	0xFFFE_0000	5.3.1.17/5-25
0x44–0x50	Reserved			
0x54	DIU Clock Config Register (DCCR)	R/W	0x0000_0000	5.3.1.18/5-26
0x58	MSCAN1 Clock Control Register (M1CCR)	R/W	0xFFFE_0000	5.3.1.19/5-27
0x5C	MSCAN2 Clock Control Register (M2CCR)	R/W	0xFFFE_0000	5.3.1.20/5-28
0x60	MSCAN3 Clock Control Register (M3CCR)	R/W	0xFFFE_0000	5.3.1.21/5-29
0x64	MSCAN4 Clock Control Register (M4CCR)	R/W	0xFFFE_0000	5.3.1.22/5-30
0x68–0x6F	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x70	OUT CLK0 Clock Configure Register (OUT0CCR)	R/W	0xFFFE_0000	5.3.1.23/5-30
0x74	OUT CLK1 Clock Configure Register (OUT1CCR)	R/W	0xFFFE_0000	5.3.1.24/5-31
0x78	OUT CLK2 Clock Configure Register (OUT2CCR)	R/W	0xFFFE_0000	5.3.1.25/5-32
0x7C	OUT CLK3 Clock Configure Register (OUT3CCR)	R/W	0xFFFE_0000	5.3.1.26/5-33
0x80	System Clock Frequency Register 3 (SCFR3)	R/W	0x2860_0000	5.3.1.27/5-34
0x84–0x8F	Reserved			
0x90	System PLL lock counter (SPLL_LOCK_CNT)	R	0x000U_UUUU	5.3.1.28/5-35
0x94–0xFF	Reserved			
Power Management Control (PMC) 0xFF40_1000[2] **Chapter 24, "Power Management Control Module (PMC)"**				
0x00	PMC Configuration Register (PMC_PMCCR)	R/W	0x0000_0000	24.2.2.1/24-2
0x04	PMC Event Register (PMC_PMCER)	R/W	0x0000_0000	24.2.2.2/24-3
0x08	PMC Mask Register (PMC_PMCMR)	R/W	0x0000_0000	24.2.2.3/24-4
0x0C	PMC CORE_PLL Shadow Register (PMC_PMCSR)	R/W	0x0000_0000	24.2.2.4/24-5
0x10	PMC Wakeup Source Enable Register (PMC_PMCWSE)	R/W	0x0000_0000	24.2.2.5/24-5
0x14	PMC Wakeup Source Polarity Register (PMC_PMCWSP)	R/W	0x0000_00FF	24.2.2.6/24-6
0x18–0xFF	Reserved			
General Purpose I/O 1 (GPIO1) 0xFF40_1100[2] **Chapter 16, "General Purpose I/O (GPIO)"**				
0x00	GPIO Direction Register (GPIO_GPDIR)	R/W	0x0000_0000	16.3.1.1/16-3
0x04	GPIO Open Drain Register (GPIO_GPODR)	R/W	0x0000_0000	16.3.1.2/16-3
0x08	GPIO Data Register (GPIO_GPDAT)	R/W	0x0000_0000	16.3.1.3/16-4
0x0C	GPIO Interrupt Event Register (GPIO_GPIER)	R/W	0x0000_0000	16.3.1.4/16-5
0x10	GPIO Interrupt Mask Register (GPIO_GPIMR)	R/W	0x0000_0000	16.3.1.5/16-5
0x14	GPIO External Interrupt Control Register 1 (GPIO_GPICR1)	R/W	0x0000_0000	16.3.1.6/16-6
0x18	GPIO External Interrupt Control Register 2 (GPIO_GPICR2)	R/W	0x0000_0000	16.3.1.6/16-6
0x1C–0x7F	Reserved			
General Purpose I/O 2 (GPIO2) 0xFF40_1180[2] **Chapter 16, "General Purpose I/O (GPIO)"**				
0x00	GPIO Direction Register (GPIO_GPDIR)	R/W	0x0000_0000	16.3.1.1/16-3
0x04	GPIO Open Drain Register (GPIO_GPODR)	R/W	0x0000_0000	16.3.1.2/16-3

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x08	GPIO Data Register (GPIO_GPDAT)	R/W	0x0000_0000	16.3.1.3/16-4
0x0C	GPIO Interrupt Event Register (GPIO_GPIER)	R/W	0x0000_0000	16.3.1.4/16-5
0x10	GPIO Interrupt Mask Register (GPIO_GPIMR)	R/W	0x0000_0000	16.3.1.5/16-5
0x14	GPIO External Interrupt Control Register 1 (GPIO_GPICR1)	R/W	0x0000_0000	16.3.1.6/16-6
0x18	GPIO External Interrupt Control Register 2 (GPIO_GPICR2)	R/W	0x0000_0000	16.3.1.6/16-6
0x1C–0x7F	Reserved			
MSCAN1 0xFF40_1300[2] **Chapter 22, "MSCAN"**				
0x00	MSCAN Control Register 0 (CANCTL0)	R/W	0x01	22.3.2.1/22-10
0x01	MSCAN Control Register 1 (CANCTL1)	R/W	0x11	22.3.2.2/22-12
0x02–0x03	Reserved			
0x04	MSCAN Bus Timing Register 0 (CANBTR0)	R/W	0x00	22.3.2.3/22-13
0x05	MSCAN Bus Timing Register 1 (CANBTR1)	R/W	0x00	22.3.2.4/22-14
0x06–0x07	Reserved			
0x08	MSCAN Receiver Flag Register (CANRFLG)	R/W	0x00	22.3.2.5/22-15
0x09	MSCAN Receiver Interrupt Enable Register (CANRIER)	R/W	0x00	22.3.2.6/22-16
0x0A–0x0B	Reserved			
0x0C	MSCAN Transmitter Flag Register (CANTFLG)	R/W	0x07	22.3.2.7/22-18
0x0D	MSCAN Transmitter Interrupt Enable Register (CANTIER)	R/W	0x00	22.3.2.8/22-18
0x0E–0x0F	Reserved			
0x10	MSCAN Transmitter Message Abort Control (CANTARQ)	R/W	0x00	22.3.2.9/22-19
0x11	MSCAN Transmitter Message Abort Control (CANTAAK)	R	0x00	22.3.2.10/22-20
0x12–0x13	Reserved			
0x14	MSCAN Transmit Buffer Selection (CANTBSEL)	R/W	0x00	22.3.2.11/22-21
0x15	MSCAN Identifier Acceptance Control Register (CANIDAC)	R/W	0x00	22.3.2.12/22-22
0x16–0x18	Reserved			
0x19	MSCAN MISC Register (CANMISC)	R/W	0x00	22.3.2.13/22-22
0x1A–0x1B	Reserved			
0x1C	MSCAN Receive Error Counter Register (CANRXERR)	R	0x00	22.3.2.14/22-23
0x1D	MSCAN Transmitter Error Counter Register (CANTXERR)	R	0x00	22.3.2.15/22-23

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x1E–0x1F	Reserved			
0x20	MSCAN Identifier Acceptance Register 0 (CANIDAR0)	R/W	0x00	22.3.2.16/22-24
0x21	MSCAN Identifier Acceptance Register 1 (CANIDAR1)	R/W	0x00	22.3.2.16/22-24
0x22–0x23	Reserved			
0x24	MSCAN Identifier Acceptance Register 2 (CANIDAR2)	R/W	0x00	22.3.2.16/22-24
0x25	MSCAN Identifier Acceptance Register 3 (CANIDAR3)	R/W	0x00	22.3.2.16/22-24
0x26–0x27	Reserved			
0x28	MSCAN Identifier Mask Register 0 (CANIDMR0)	R/W	0x00	22.3.2.17/22-25
0x29	MSCAN Identifier Mask Register 1 (CANIDMR1)	R/W	0x00	22.3.2.17/22-25
0x2A–0x2B	Reserved			
0x2C	MSCAN Identifier Mask Register 2 (CANIDMR2)	R/W	0x00	22.3.2.17/22-25
0x2D	MSCAN Identifier Mask Register 3 (CANIDMR3)	R/W	0x00	22.3.2.17/22-25
0x2E–0x2F	Reserved			
0x30	MSCAN Identifier Acceptance Register 4 (CANIDAR4)	R/W	0x00	22.3.2.17/22-25
0x31	MSCAN Identifier Acceptance Register 5 (CANIDAR5)	R/W	0x00	22.3.2.17/22-25
0x32–0x33	Reserved			
0x34	MSCAN Identifier Acceptance Register 6 (CANIDAR6)	R/W	0x00	22.3.2.17/22-25
0x35	MSCAN Identifier Acceptance Register 7 (CANIDAR7)	R/W	0x00	22.3.2.17/22-25
0x36–0x37	Reserved			
0x38	MSCAN Identifier Mask Register 4 (CANIDMR4)	R/W	0x00	22.3.2.17/22-25
0x39	MSCAN Identifier Mask Register 5 (CANIDMR5)	R/W	0x00	22.3.2.17/22-25
0x3A–0x3B	Reserved			
0x3C	MSCAN Identifier Mask Register 6 (CANIDMR6)	R/W	0x00	22.3.2.17/22-25
0x3D	MSCAN Identifier Mask Register 7 (CANIDMR7)	R/W	0x00	22.3.2.17/22-25
0x3E–0x3F	Reserved			
0x40–0x5F	**MSCAN Receive Message Buffer (CANRXFG)**			
0x40	RX Identifier Register 0 (IDR0)	R/W	—[3]	22.3.3.1/22-28
0x41	RX Identifier Register 1 (IDR1)	R/W	—[3]	22.3.3.1/22-28
0x42–0x43	Reserved			
0x44	RX Identifier Register 2 (IDR2)	R/W	—[3]	22.3.3.1/22-28
0x45	RX Identifier Register 3 (IDR3)	R/W	—[3]	22.3.3.1/22-28
0x46–0x47	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x48	RX Data Segment Register 0 (DSR0)	R/W	—[3]	22.3.3.2/22-31
0x49	RX Data Segment Register 1 (DSR1)	R/W	—[3]	22.3.3.2/22-31
0x4A–0x4B	Reserved			
0x4C	RX Data Segment Register 2 (DSR2)	R/W	—[3]	22.3.3.2/22-31
0x4D	RX Data Segment Register 3 (DSR3)	R/W	—[3]	22.3.3.2/22-31
0x4E–0x4F	Reserved			
0x50	RX Data Segment Register 4 (DSR4)	R/W	—[3]	22.3.3.2/22-31
0x51	RX Data Segment Register 5 (DSR5)	R/W	—[3]	22.3.3.2/22-31
0x52–0x53	Reserved			
0x54	RX Data Segment Register 6 (DSR6)	R/W	—[3]	22.3.3.2/22-31
0x55	RX Data Segment Register 7 (DSR7)	R/W	—[3]	22.3.3.2/22-31
0x56–0x57	Reserved			
0x58	RX Data Length Register (DLR)	R/W	—[3]	22.3.3.3/22-32
0x59	RX Transmit Buffer Priority Register (TBPR)	R/W	0x00	22.3.3.4/22-32
0x5A–0x5B	Reserved			
0x5C	RX Time Stamp Register High (TSRH)	R	—[3]	22.3.3.5/22-33
0x5D	RX Time Stamp Register Low (TSRL)	R	—[3]	22.3.3.5/22-33
0x5E–0x5F	Reserved			
0x60–0x7F	**MSCAN Transmit Message Buffer (CANTXFG)**			
0x60	TX Identifier Register 0 (IDR0)	R/W	—[3]	22.3.3.1/22-28
0x61	TX Identifier Register 1 (IDR1)	R/W	—[3]	22.3.3.1/22-28
0x62–0x63	Reserved			
0x64	TX Identifier Register 2 (IDR2)	R/W	—[3]	22.3.3.1/22-28
0x65	TX Identifier Register 3 (IDR3)	R/W	—[3]	22.3.3.1/22-28
0x66–0x67	Reserved			
0x68	TX Data Segment Register 0 (DSR0)	R/W	—[3]	22.3.3.2/22-31
0x69	TX Data Segment Register 1 (DSR1)	R/W	—[3]	22.3.3.2/22-31
0x6A–0x6B	Reserved			
0x6C	TX Data Segment Register 2 (DSR2)	R/W	—[3]	22.3.3.2/22-31
0x6D	TX Data Segment Register 3 (DSR3)	R/W	—[3]	22.3.3.2/22-31
0x6E–0x6F	Reserved			
0x70	TX Data Segment Register 4 (DSR4)	R/W	—[3]	22.3.3.2/22-31

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x71	TX Data Segment Register 5 (DSR5)	R/W	—[3]	22.3.3.2/22-31
0x72–0x73	Reserved			
0x74	TX Data Segment Register 6 (DSR6)	R/W	—[3]	22.3.3.2/22-31
0x75	TX Data Segment Register 7 (DSR7)	R/W	—[3]	22.3.3.2/22-31
0x76–0x77	Reserved			
0x78	TX Data Length Register (DLR)	R/W	—[3]	22.3.3.3/22-32
0x79	TX Transmit Buffer Priority Register (TBPR)	R/W	0x00	22.3.3.4/22-32
0x7A–0x7B	Reserved			
0x7C	TX Time Stamp Register High (TSRH)	R	—[3]	22.3.3.5/22-33
0x7D	TX Time Stamp Register Low (TSRL)	R	—[3]	22.3.3.5/22-33
0x7E–0x7F	Reserved			
MSCAN2 0xFF40_1380[2] **Chapter 22, “MSCAN”**				
0x00	MSCAN Control Register 0 (CANCTL0)	R/W	0x01	22.3.2.1/22-10
0x01	MSCAN Control Register 1 (CANCTL1)	R/W	0x11	22.3.2.2/22-12
0x02–0x03	Reserved			
0x04	MSCAN Bus Timing Register 0 (CANBTR0)	R/W	0x00	22.3.2.3/22-13
0x05	MSCAN Bus Timing Register 1 (CANBTR1)	R/W	0x00	22.3.2.4/22-14
0x06–0x07	Reserved			
0x08	MSCAN Receiver Flag Register (CANRFLG)	R/W	0x00	22.3.2.5/22-15
0x09	MSCAN Receiver Interrupt Enable Register (CANRIER)	R/W	0x00	22.3.2.6/22-16
0x0A–0x0B	Reserved			
0x0C	MSCAN Transmitter Flag Register (CANTFLG)	R/W	0x07	22.3.2.7/22-18
0x0D	MSCAN Transmitter Interrupt Enable Register (CANTIER)	R/W	0x00	22.3.2.8/22-18
0x0E–0x0F	Reserved			
0x10	MSCAN Transmitter Message Abort Control (CANTARQ)	R/W	0x00	22.3.2.9/22-19
0x11	MSCAN Transmitter Message Abort Control (CANTAAK)	R	0x00	22.3.2.10/22-20
0x12–0x13	Reserved			
0x14	MSCAN Transmit Buffer Selection (CANTBSEL)	R/W	0x00	22.3.2.11/22-21
0x15	MSCAN Identifier Acceptance Control Register (CANIDAC)	R/W	0x00	22.3.2.12/22-22
0x16–0x18	Reserved			
0x19	MSCAN MISC Register (CANMISC)	R/W	0x00	22.3.2.13/22-22

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x1A–0x1B	Reserved			
0x1C	MSCAN Receive Error Counter Register (CANRXERR)	R	0x00	22.3.2.14/22-23
0x1D	MSCAN Transmitter Error Counter Register (CANTXERR)	R	0x00	22.3.2.15/22-23
0x1E–0x1F	Reserved			
0x20	MSCAN Identifier Acceptance Registers (CANIDAR0)	R/W	0x00	22.3.2.16/22-24
0x21	MSCAN Identifier Acceptance Registers (CANIDAR1)	R/W	0x00	22.3.2.16/22-24
0x22–0x23	Reserved			
0x24	MSCAN Identifier Acceptance Registers (CANIDAR2)	R/W	0x00	22.3.2.16/22-24
0x25	MSCAN Identifier Acceptance Registers (CANIDAR3)	R/W	0x00	22.3.2.16/22-24
0x26–0x27	Reserved			
0x28	MSCAN Identifier Mask Registers (CANIDMR0)	R/W	0x00	22.3.2.17/22-25
0x29	MSCAN Identifier Mask Registers (CANIDMR1)	R/W	0x00	22.3.2.17/22-25
0x2A–0x2B	Reserved			
0x2C	MSCAN Identifier Mask Registers (CANIDMR2)	R/W	0x00	22.3.2.17/22-25
0x2D	MSCAN Identifier Mask Registers (CANIDMR3)	R/W	0x00	22.3.2.17/22-25
0x2E–0x2F	Reserved			
0x30	MSCAN Identifier Acceptance Registers (CANIDAR4)	R/W	0x00	22.3.2.17/22-25
0x31	MSCAN Identifier Acceptance Registers (CANIDAR5)	R/W	0x00	22.3.2.17/22-25
0x32–0x33	Reserved			
0x34	MSCAN Identifier Acceptance Registers (CANIDAR6)	R/W	0x00	22.3.2.17/22-25
0x35	MSCAN Identifier Acceptance Registers (CANIDAR7)	R/W	0x00	22.3.2.17/22-25
0x36–0x37	Reserved			
0x38	MSCAN Identifier Mask Registers (CANIDMR4)	R/W	0x00	22.3.2.17/22-25
0x39	MSCAN Identifier Mask Registers (CANIDMR5)	R/W	0x00	22.3.2.17/22-25
0x3A–0x3B	Reserved			
0x3C	MSCAN Identifier Mask Registers (CANIDMR6)	R/W	0x00	22.3.2.17/22-25
0x3D	MSCAN Identifier Mask Registers (CANIDMR7)	R/W	0x00	22.3.2.17/22-25
0x3E–0x3F	Reserved			
0x40–0x5F	**MSCAN Receive Message Buffer (CANRXFG)**			
0x40	RX Identifier Register 0 (IDR0)	R/W	—[3]	22.3.3.1/22-28
0x41	RX Identifier Register 1 (IDR1)	R/W	—[3]	22.3.3.1/22-28

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x42–0x43	Reserved			
0x44	RX Identifier Register 2 (IDR2)	R/W	—[3]	22.3.3.1/22-28
0x45	RX Identifier Register 3 (IDR3)	R/W	—[3]	22.3.3.1/22-28
0x46–0x47	Reserved			
0x48	RX Data Segment Register 0 (DSR0)	R/W	—[3]	22.3.3.2/22-31
0x49	RX Data Segment Register 1 (DSR1)	R/W	—[3]	22.3.3.2/22-31
0x4A–0x4B	Reserved			
0x4C	RX Data Segment Register 2 (DSR2)	R/W	—[3]	22.3.3.2/22-31
0x4D	RX Data Segment Register 3 (DSR3)	R/W	—[3]	22.3.3.2/22-31
0x4E–0x4F	Reserved			
0x50	RX Data Segment Register 4 (DSR4)	R/W	—[3]	22.3.3.2/22-31
0x51	RX Data Segment Register 5 (DSR5)	R/W	—[3]	22.3.3.2/22-31
0x52–0x53	Reserved			
0x54	RX Data Segment Register 6 (DSR6)	R/W	—[3]	22.3.3.2/22-31
0x55	RX Data Segment Register 7 (DSR7)	R/W	—[3]	22.3.3.2/22-31
0x56–0x57	Reserved			
0x58	RX Data Length Register (DLR)	R/W	—[3]	22.3.3.3/22-32
0x59	RX Transmit Buffer Priority Register (TBPR)	R/W	0x00	22.3.3.4/22-32
0x5A–0x5B	Reserved			
0x5C	RX Time Stamp Register High (TSRH)	R	—[3]	22.3.3.5/22-33
0x5D	RX Time Stamp Register Low (TSRL)	R	—[3]	22.3.3.5/22-33
0x5E–0x5F	Reserved			
0x60–0x7F	**MSCAN Transmit Message Buffer (CANTXFG)**			
0x60	TX Identifier Register 0 (IDR0)	R/W	—[3]	22.3.3.1/22-28
0x61	TX Identifier Register 1 (IDR1)	R/W	—[3]	22.3.3.1/22-28
0x62–0x63	Reserved			
0x64	TX Identifier Register 2 (IDR2)	R/W	—[3]	22.3.3.1/22-28
0x65	TX Identifier Register 3 (IDR3)	R/W	—[3]	22.3.3.1/22-28
0x66–0x67	Reserved			
0x68	TX Data Segment Register 0 (DSR0)	R/W	—[3]	22.3.3.2/22-31
0x69	TX Data Segment Register 1 (DSR1)	R/W	—[3]	22.3.3.2/22-31
0x6A–0x6B	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x6C	TX Data Segment Register 2 (DSR2)	R/W	—[3]	22.3.3.2/22-31
0x6D	TX Data Segment Register 3 (DSR3)	R/W	—[3]	22.3.3.2/22-31
0x6E–0x6F	Reserved			
0x70	TX Data Segment Register 4 (DSR4)	R/W	—[3]	22.3.3.2/22-31
0x71	TX Data Segment Register 5 (DSR5)	R/W	—[3]	22.3.3.2/22-31
0x72–0x73	Reserved			
0x74	TX Data Segment Register 6 (DSR6)	R/W	—[3]	22.3.3.2/22-31
0x75	TX Data Segment Register 7 (DSR7)	R/W	—[3]	22.3.3.2/22-31
0x76–0x77	Reserved			
0x78	TX Data Length Register (DLR)	R/W	—[3]	22.3.3.3/22-32
0x79	TX Transmit Buffer Priority Register (TBPR)	R/W	0x00	22.3.3.4/22-32
0x7A–0x7B	Reserved			
0x7C	TX Time Stamp Register High (TSRH)	R	—[3]	22.3.3.5/22-33
0x7D	TX Time Stamp Register Low (TSRL)	R	—[3]	22.3.3.5/22-33
0x7E–0x7F	Reserved			
Byte Data Link Controller (BDLC) 0xFF40_1400[2] **Chapter 6, "Byte Data Link Controller (BDLC)"**				
0x00	BDLC Control Register 1 (BDLC_DLCBCR1)	R/W	0xC0	6.3.2.1/6-5
0x01	BDLC State Vector Register (BDLC_DLCBSVR)	R	0x00	6.3.2.2/6-6
0x02–0x03	Reserved			
0x04	BDLC Control Register 2 (BDLC_DLCBCR2)	R/W	0x40	6.3.2.3/6-8
0x05	BDLC Data Register (BDLC_DLCBDR)	R/W	0x00	6.3.2.4/6-13
0x06–0x07	Reserved			
0x08	BDLC Analog Round Trip Delay Register (BDLC_DLCBARD)	R/W	0x50	6.3.2.5/6-14
0x09	BDLC Rate Select Register (BDLC_DLCBRSR)	R/W	0x00	6.3.2.6/6-16
0x0A–0x0B	Reserved			
0x0C	BDLC Control Register (BDLC_DLCSCR)	R/W	0x00	6.3.2.7/6-17
0x0D	BDLC Status Register (BDLC_DLCBSTAT)	R/W	0x00	6.3.2.8/6-18
0x0E–0xFF	Reserved			
Secure Digital Host Controller (SDHC1) 0xFF40_1500[2] **Chapter 28, "Secure Digital Host Controller (SDHC)"**				
0x00	SDHC_STR_STP_CLK—SDHC Clock Control register	R/W	0x0000_0000	28.3.2.1/28-4

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x04	SDHC_STATUS—SDHC Status register	R/W	0x0000_0000	28.3.2.2/28-5
0x08	SDHC_CLK_RATE—SDHC Card Clock Rate register	R/W	0x0000_0008	28.3.2.3/28-9
0x0C	SDHC_CMD_DAT_CONT—SDHC Command and Data Control register	R/W	0x0000_0000	28.3.2.4/28-10
0x10	SDHC_RES_TO—SDHC Response Time-out register	R/W	0x0000_0040	28.3.2.5/28-12
0x14	SDHC_READ_TO—SDHC Read Time-out register	R/W	0x0000_FFFF	28.3.2.6/28-13
0x18	SDHC_BLK_LEN—SDHC Block Length register	R/W	0x0000_0000	28.3.2.7/28-14
0x1C	SDHC_NOB—SDHC Number of Block register	R/W	0x0000_0000	28.3.2.8/28-15
0x20	SDHC_REV_NO—SDHC Revision Number register	R	0x0000_0400	28.3.2.9/28-16
0x24	SDHC_INT_CNTR—SDHC Interrupt Control register	R/W	0x0000_0000	28.3.2.10/28-17
0x28	SDHC_CMD—SDHC Command Number register	R/W	0x0000_0000	28.3.2.11/28-20
0x2C	SDHC_ARG—SDHC Command Argument register	R/W	0x0000_0000	28.3.2.12/28-21
0x30	Reserved			
0x34	SDHC_RES_FIFO—SDHC Command Response FIFO Access register	R	0x0000_0000	28.3.2.13/28-21
0x38	SDHC_BUFFER_ACCESS—SDHC Data Buffer Access register	R/W	0x0000_0000	28.3.2.14/28-22
0x3C–0xFF	Reserved			
Inter-Integrated Circuit 1 (I^2C1) 0xFF40_1700[2] **Chapter 19, "Inter-Integrated Circuit (I2C)"**				
0x00	I2C_MADR1 – I^2C1 Address Register	R/W	0x0000_0000	19.3.1.1/19-9
0x04	I2C_MFDR1 – I^2C1 Frequency Divider Register	R/W	0x0000_0000	19.3.1.2/19-9
0x08	I2C_MCR1 – I^2C1 Control Register	R/W	0x0000_0000	19.3.1.3/19-16
0x0C	I2C_MSR1 – I^2C1 Status Register	R/W	0x8000_0000	19.3.1.4/19-17
0x10	I2C_MDR1 – I^2C1 Date I/O Register	R/W	0x0000_0000	19.3.1.5/19-20
0x20	I2C_MADR2 – I^2C2 Address Register	R/W	0x0000_0000	19.3.1.1/19-9
0x24	I2C_MFDR2 – I^2C2 Frequency Divider Register	R/W	0x0000_0000	19.3.1.2/19-9
0x28	I2C_MCR2 – I^2C2 Control Register	R/W	0x0000_0000	19.3.1.3/19-16
0x2C	I2C_MSR2 – I^2C2 Status Register	R/W	0x8000_0000	19.3.1.4/19-17
0x30	I2C_MDR2 – I^2C2 Date I/O Register	R/W	0x0000_0000	19.3.1.5/19-20
0x40	I2C_MADR3 – I^2C3 Address Register	R/W	0x0000_0000	19.3.1.1/19-9
0x44	I2C_MFDR3 – I^2C3 Frequency Divider Register	R/W	0x0000_0000	19.3.1.2/19-9
0x48	I2C_MCR3 – I^2C3 Control Register	R/W	0x0000_0000	19.3.1.3/19-16

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x4C	I2C_MSR3 – I²C3 Status Register	R/W	0x8000_0000	19.3.1.4/19-17
0x50	I2C_MDR3 – I²C3 Date I/O Register	R/W	0x0000_0000	19.3.1.5/19-20
0x54–0x5F	Reserved			
0x60	I2C_ICR – I²C Interrupt Control Register	R/W	0x1300_0000	19.3.1.6/19-20
0x64	I2C_MIFR – I²C Filter Register	R/W	0x0000_0000	19.3.1.7/19-21
0x68–0xFF	Reserved			
Inter-Integrated Circuit 2 (I²C2) 0xFF40_1720[2] **Chapter 19, "Inter-Integrated Circuit (I2C)"**				
0x00	I2C_MADR1 – I²C1 Address Register	R/W	0x0000_0000	19.3.1.1/19-9
0x04	I2C_MFDR1 – I²C1 Frequency Divider Register	R/W	0x0000_0000	19.3.1.2/19-9
0x08	I2C_MCR1 – I²C1 Control Register	R/W	0x0000_0000	19.3.1.3/19-16
0x0C	I2C_MSR1 – I²C1 Status Register	R/W	0x8000_0000	19.3.1.4/19-17
0x10	I2C_MDR1 – I²C1 Date I/O Register	R/W	0x0000_0000	19.3.1.5/19-20
0x20	I2C_MADR2 – I²C2 Address Register	R/W	0x0000_0000	19.3.1.1/19-9
0x24	I2C_MFDR2 – I²C2 Frequency Divider Register	R/W	0x0000_0000	19.3.1.2/19-9
0x28	I2C_MCR2 – I²C2 Control Register	R/W	0x0000_0000	19.3.1.3/19-16
0x2C	I2C_MSR2 – I²C2 Status Register	R/W	0x8000_0000	19.3.1.4/19-17
0x30	I2C_MDR2 – I²C2 Date I/O Register	R/W	0x0000_0000	19.3.1.5/19-20
0x40	I2C_MADR3 – I²C3 Address Register	R/W	0x0000_0000	19.3.1.1/19-9
0x44	I2C_MFDR3 – I²C3 Frequency Divider Register	R/W	0x0000_0000	19.3.1.2/19-9
0x48	I2C_MCR3 – I²C3 Control Register	R/W	0x0000_0000	19.3.1.3/19-16
0x4C	I2C_MSR3 – I²C3 Status Register	R/W	0x8000_0000	19.3.1.4/19-17
0x50	I2C_MDR3 – I²C3 Date I/O Register	R/W	0x0000_0000	19.3.1.5/19-20
0x54–0x5F	Reserved			
0x60	I2C_ICR – I²C Interrupt Control Register	R/W	0x1300_0000	19.3.1.6/19-20
0x64	I2C_MIFR – I²C Filter Register	R/W	0x0000_0000	19.3.1.7/19-21
0x68–0xFF	Reserved			
Inter-Integrated Circuit 3 (I²C3) 0xFF40_1740[2] **Chapter 19, "Inter-Integrated Circuit (I2C)"**				
0x00	I2C_MADR1 – I²C1 Address Register	R/W	0x0000_0000	19.3.1.1/19-9
0x04	I2C_MFDR1 – I²C1 Frequency Divider Register	R/W	0x0000_0000	19.3.1.2/19-9
0x08	I2C_MCR1 – I²C1 Control Register	R/W	0x0000_0000	19.3.1.3/19-16

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x0C	I2C_MSR1 – I^{2}C1 Status Register	R/W	0x8000_0000	19.3.1.4/19-17
0x10	I2C_MDR1 – I^{2}C1 Date I/O Register	R/W	0x0000_0000	19.3.1.5/19-20
0x20	I2C_MADR2 – I^{2}C2 Address Register	R/W	0x0000_0000	19.3.1.1/19-9
0x24	I2C_MFDR2 – I^{2}C2 Frequency Divider Register	R/W	0x0000_0000	19.3.1.2/19-9
0x28	I2C_MCR2 – I^{2}C2 Control Register	R/W	0x0000_0000	19.3.1.3/19-16
0x2C	I2C_MSR2 – I^{2}C2 Status Register	R/W	0x8000_0000	19.3.1.4/19-17
0x30	I2C_MDR2 – I^{2}C2 Date I/O Register	R/W	0x0000_0000	19.3.1.5/19-20
0x40	I2C_MADR3 – I^{2}C3 Address Register	R/W	0x0000_0000	19.3.1.1/19-9
0x44	I2C_MFDR3 – I^{2}C3 Frequency Divider Register	R/W	0x0000_0000	19.3.1.2/19-9
0x48	I2C_MCR3 – I^{2}C3 Control Register	R/W	0x0000_0000	19.3.1.3/19-16
0x4C	I2C_MSR3 – I^{2}C3 Status Register	R/W	0x8000_0000	19.3.1.4/19-17
0x50	I2C_MDR3 – I^{2}C3 Date I/O Register	R/W	0x0000_0000	19.3.1.5/19-20
0x54–0x5F	Reserved			
0x60	I2C_ICR – I^{2}C Interrupt Control Register	R/W	0x1300_0000	19.3.1.6/19-20
0x64	I2C_MIFR – I^{2}C Filter Register	R/W	0x0000_0000	19.3.1.7/19-21
0x68–0xFF	Reserved			
Display Interface Unit (DIU) 0xFF40_2100[2] **Chapter 10, "Display Interface Unit (DIU)"**				
0x00	DESC_1—Plane 1 Area Descriptor Pointer Register	R/W	0x0000_0000	10.3.3.1/10-8
0x04	DESC_2—Plane 2 Area Descriptor Pointer Register	R/W	0x0000_0000	10.3.3.2/10-8
0x08	DESC_3—Plane 3 Area Descriptor Pointer Register	R/W	0x0000_0000	10.3.3.3/10-9
0x0C	GAMMA—Gamma Table Pointer Register	R/W	0x0000_0000	10.3.3.4/10-9
0x10	PALETTE—Palette Table Pointer Register	R/W	0x0000_0000	10.3.3.5/10-10
0x14	CURSOR—Cursor Bitmap Table Pointer Register	R/W	0x0000_0000	10.3.3.6/10-11
0x18	CURS_POS—Cursor Position Register	R/W	0x0000_0000	10.3.3.7/10-11
0x1C	DIU_MODE—DIU Mode of Operation Register	R/W	0x0000_0000	10.3.3.8/10-12
0x20	BGND—Background Color Register	R/W	0x0000_0000	10.3.3.9/10-12
0x24	BGND_WB—Background Color in Writeback Mode Register	R/W	0x0000_0000	10.3.3.10/10-13
0x28	DISP_SIZE—Display Size Register	R/W	0x0000_0000	10.3.3.11/10-14
0x2C	WB_SIZE—Writeback Plane Size Register	R/W	0x0000_0000	10.3.3.12/10-14
0x30	WB_MEM_ADDR—Writeback Plane Address Register	R/W	0x0000_0000	10.3.3.13/10-15

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x34	HSYN_PARA—Horizontal Sync Pulse Parameters Register	R/W	0x0000_0000	10.3.3.14/10-16
0x38	VSYN_PARA—Vertical Sync Pulse Parameters Register	R/W	0x0000_0000	10.3.3.15/10-16
0x3C	SYN_POL— Synchronization Signals Polarity Register	R/W	0x0000_0000	10.3.3.16/10-17
0x40	THRESHOLDS—Thresholds Register	R/W	0x8000_F800	10.3.3.17/10-18
0x44	INT_STATUS—Interrupt Status Register	R	0x0000_0000	10.3.3.18/10-18
0x48	INT_MASK—Interrupt Mask Register	R/W	0x0000_003F	10.3.3.19/10-19
0x4C	COLBAR_1—Color #1 in the Color Bar (Black) Register	R/W	0xFF00_0000	10.3.3.20.1/10-21
0x50	COLBAR_2—Color #2 in the Color Bar (Blue) Register	R/W	0xFF00_00FF	10.3.3.20.2/10-21
0x54	COLBAR_3—Color #3 in the Color Bar (Cyan) Register	R/W	0xFF00_FFFF	10.3.3.20.3/10-21
0x58	COLBAR_4—Color #4 in the Color Bar (Green) Register	R/W	0xFF00_FF00	10.3.3.20.4/10-22
0x5C	COLBAR_5—Color #5 in the Color Bar (Yellow) Register	R/W	0xFFFF_FF00	10.3.3.20.5/10-22
0x60	COLBAR_6—Color #6 in the Color Bar (Red) Register	R/W	0xFFFF_0000	10.3.3.20.6/10-22
0x64	COLBAR_7—Color #7in the Color Bar (Purple) Register	R/W	0xFFFF_00FF	10.3.3.20.7/10-23
0x68	COLBAR_8—Color #8 in the Color Bar (White) Register	R/W	0xFFFF_FFFF	10.3.3.20.8/10-23
0x6C	FILLING—Filling Status Register	R	0x0000_0000	10.3.3.21/10-23
0x70	PLUT—Priority Look Up Table Register	R/W	0x0000_0000	10.3.3.22/10-24
0x74–0xFF	Reserved			
MSCAN3 0xFF40_2300[2] **Chapter 22, "MSCAN"**				
0x00	MSCAN Control Register 0 (CANCTL0)	R/W	0x01	22.3.2.1/22-10
0x01	MSCAN Control Register 1 (CANCTL1)	R/W	0x11	22.3.2.2/22-12
0x02–0x03	Reserved			
0x04	MSCAN Bus Timing Register 0 (CANBTR0)	R/W	0x00	22.3.2.3/22-13
0x05	MSCAN Bus Timing Register 1 (CANBTR1)	R/W	0x00	22.3.2.4/22-14
0x06–0x07	Reserved			
0x08	MSCAN Receiver Flag Register (CANRFLG)	R/W	0x00	22.3.2.5/22-15
0x09	MSCAN Receiver Interrupt Enable Register (CANRIER)	R/W	0x00	22.3.2.6/22-16
0x0A–0x0B	Reserved			
0x0C	MSCAN Transmitter Flag Register (CANTFLG)	R/W	0x07	22.3.2.7/22-18
0x0D	MSCAN Transmitter Interrupt Enable Register (CANTIER)	R/W	0x00	22.3.2.8/22-18
0x0E–0x0F	Reserved			
0x10	MSCAN Transmitter Message Abort Control (CANTARQ)	R/W	0x00	22.3.2.9/22-19

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x11	MSCAN Transmitter Message Abort Control (CANTAAK)	R	0x00	22.3.2.10/22-20
0x12–0x13	Reserved			
0x14	MSCAN Transmit Buffer Selection (CANTBSEL)	R/W	0x00	22.3.2.11/22-21
0x15	MSCAN Identifier Acceptance Control Register (CANIDAC)	R/W	0x00	22.3.2.12/22-22
0x16–0x18	Reserved			
0x19	MSCAN MISC Register (CANMISC)	R/W	0x00	22.3.2.13/22-22
0x1A–0x1B	Reserved			
0x1C	MSCAN Receive Error Counter Register (CANRXERR)	R	0x00	22.3.2.14/22-23
0x1D	MSCAN Transmitter Error Counter Register (CANTXERR)	R	0x00	22.3.2.15/22-23
0x1E–0x1F	Reserved			
0x20	MSCAN Identifier Acceptance Registers (CANIDAR0)	R/W	0x00	22.3.2.16/22-24
0x21	MSCAN Identifier Acceptance Registers (CANIDAR1)	R/W	0x00	22.3.2.16/22-24
0x22–0x23	Reserved			
0x24	MSCAN Identifier Acceptance Registers (CANIDAR2)	R/W	0x00	22.3.2.16/22-24
0x25	MSCAN Identifier Acceptance Registers (CANIDAR3)	R/W	0x00	22.3.2.16/22-24
0x26–0x27	Reserved			
0x28	MSCAN Identifier Mask Registers (CANIDMR0)	R/W	0x00	22.3.2.17/22-25
0x29	MSCAN Identifier Mask Registers (CANIDMR1)	R/W	0x00	22.3.2.17/22-25
0x2A–0x2B	Reserved			
0x2C	MSCAN Identifier Mask Registers (CANIDMR2)	R/W	0x00	22.3.2.17/22-25
0x2D	MSCAN Identifier Mask Registers (CANIDMR3)	R/W	0x00	22.3.2.17/22-25
0x2E–0x2F	Reserved			
0x30	MSCAN Identifier Acceptance Registers (CANIDAR4)	R/W	0x00	22.3.2.17/22-25
0x31	MSCAN Identifier Acceptance Registers (CANIDAR5)	R/W	0x00	22.3.2.17/22-25
0x32–0x33	Reserved			
0x34	MSCAN Identifier Acceptance Registers (CANIDAR6)	R/W	0x00	22.3.2.17/22-25
0x35	MSCAN Identifier Acceptance Registers (CANIDAR7)	R/W	0x00	22.3.2.17/22-25
0x36–0x37	Reserved			
0x38	MSCAN Identifier Mask Registers (CANIDMR4)	R/W	0x00	22.3.2.17/22-25
0x39	MSCAN Identifier Mask Registers (CANIDMR5)	R/W	0x00	22.3.2.17/22-25
0x3A–0x3B	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x3C	MSCAN Identifier Mask Registers (CANIDMR6)	R/W	0x00	22.3.2.17/22-25
0x3D	MSCAN Identifier Mask Registers (CANIDMR7)	R/W	0x00	22.3.2.17/22-25
0x3E–0x3F	Reserved			
0x40–0x5F	**MSCAN Receive Message Buffer (CANRXFG)**			
0x40	RX Identifier Register 0 (IDR0)	R/W	—[3]	22.3.3.1/22-28
0x41	RX Identifier Register 1 (IDR1)	R/W	—[3]	22.3.3.1/22-28
0x42–0x43	Reserved			
0x44	RX Identifier Register 2 (IDR2)	R/W	—[3]	22.3.3.1/22-28
0x45	RX Identifier Register 3 (IDR3)	R/W	—[3]	22.3.3.1/22-28
0x46–0x47	Reserved			
0x48	RX Data Segment Register 0 (DSR0)	R/W	—[3]	22.3.3.2/22-31
0x49	RX Data Segment Register 1 (DSR1)	R/W	—[3]	22.3.3.2/22-31
0x4A–0x4B	Reserved			
0x4C	RX Data Segment Register 2 (DSR2)	R/W	—[3]	22.3.3.2/22-31
0x4D	RX Data Segment Register 3 (DSR3)	R/W	—[3]	22.3.3.2/22-31
0x4E–0x4F	Reserved			
0x50	RX Data Segment Register 4 (DSR4)	R/W	—[3]	22.3.3.2/22-31
0x51	RX Data Segment Register 5 (DSR5)	R/W	—[3]	22.3.3.2/22-31
0x52–0x53	Reserved			
0x54	RX Data Segment Register 6 (DSR6)	R/W	—[3]	22.3.3.2/22-31
0x55	RX Data Segment Register 7 (DSR7)	R/W	—[3]	22.3.3.2/22-31
0x56–0x57	Reserved			
0x58	RX Data Length Register (DLR)	R/W	—[3]	22.3.3.3/22-32
0x59	RX Transmit Buffer Priority Register (TBPR)	R/W	0x00	22.3.3.4/22-32
0x5A–0x5B	Reserved			
0x5C	RX Time Stamp Register High (TSRH)	R	—[3]	22.3.3.5/22-33
0x5D	RX Time Stamp Register Low (TSRL)	R	—[3]	22.3.3.5/22-33
0x5E–0x5F	Reserved			
0x60–0x7F	**MSCAN Transmit Message Buffer (CANTXFG)**			
0x60	TX Identifier Register 0 (IDR0)	R/W	—[3]	22.3.3.1/22-28
0x61	TX Identifier Register 1 (IDR1)	R/W	—[3]	22.3.3.1/22-28
0x62–0x63	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x64	TX Identifier Register 2 (IDR2)	R/W	—[3]	22.3.3.1/22-28
0x65	TX Identifier Register 3 (IDR3)	R/W	—[3]	22.3.3.1/22-28
0x66–0x67	Reserved			
0x68	TX Data Segment Register 0 (DSR0)	R/W	—[3]	22.3.3.2/22-31
0x69	TX Data Segment Register 1 (DSR1)	R/W	—[3]	22.3.3.2/22-31
0x6A–0x6B	Reserved			
0x6C	TX Data Segment Register 2 (DSR2)	R/W	—[3]	22.3.3.2/22-31
0x6D	TX Data Segment Register 3 (DSR3)	R/W	—[3]	22.3.3.2/22-31
0x6E–0x6F	Reserved			
0x70	TX Data Segment Register 4 (DSR4)	R/W	—[3]	22.3.3.2/22-31
0x71	TX Data Segment Register 5 (DSR5)	R/W	—[3]	22.3.3.2/22-31
0x72–0x73	Reserved			
0x74	TX Data Segment Register 6 (DSR6)	R/W	—[3]	22.3.3.2/22-31
0x75	TX Data Segment Register 7 (DSR7)	R/W	—[3]	22.3.3.2/22-31
0x76–0x77	Reserved			
0x78	TX Data Length Register (DLR)	R/W	—[3]	22.3.3.3/22-32
0x79	TX Transmit Buffer Priority Register (TBPR)	R/W	0x00	22.3.3.4/22-32
0x7A–0x7B	Reserved			
0x7C	TX Time Stamp Register High (TSRH)	R	—[3]	22.3.3.5/22-33
0x7D	TX Time Stamp Register Low (TSRL)	R	—[3]	22.3.3.5/22-33
0x7E–0x7F	Reserved			
MSCAN4 0xFF40_2380[2] **Chapter 22, “MSCAN”**				
0x00	MSCAN Control Register 0 (CANCTL0)	R/W	0x01	22.3.2.1/22-10
0x01	MSCAN Control Register 1 (CANCTL1)	R/W	0x11	22.3.2.2/22-12
0x02–0x03	Reserved			
0x04	MSCAN Bus Timing Register 0 (CANBTR0)	R/W	0x00	22.3.2.3/22-13
0x05	MSCAN Bus Timing Register 1 (CANBTR1)	R/W	0x00	22.3.2.4/22-14
0x06–0x07	Reserved			
0x08	MSCAN Receiver Flag Register (CANRFLG)	R/W	0x00	22.3.2.5/22-15
0x09	MSCAN Receiver Interrupt Enable Register (CANRIER)	R/W	0x00	22.3.2.6/22-16
0x0A–0x0B	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x0C	MSCAN Transmitter Flag Register (CANTFLG)	R/W	0x07	22.3.2.7/22-18
0x0D	MSCAN Transmitter Interrupt Enable Register (CANTIER)	R/W	0x00	22.3.2.8/22-18
0x0E–0x0F	Reserved			
0x10	MSCAN Transmitter Message Abort Control (CANTARQ)	R/W	0x00	22.3.2.9/22-19
0x11	MSCAN Transmitter Message Abort Control (CANTAAK)	R	0x00	22.3.2.10/22-20
0x12–0x13	Reserved			
0x14	MSCAN Transmit Buffer Selection (CANTBSEL)	R/W	0x00	22.3.2.11/22-21
0x15	MSCAN Identifier Acceptance Control Register (CANIDAC)	R/W	0x00	22.3.2.12/22-22
0x16–0x18	Reserved			
0x19	MSCAN MISC Register (CANMISC)	R/W	0x00	22.3.2.13/22-22
0x1A–0x1B	Reserved			
0x1C	MSCAN Receive Error Counter Register (CANRXERR)	R	0x00	22.3.2.14/22-23
0x1D	MSCAN Transmitter Error Counter Register (CANTXERR)	R	0x00	22.3.2.15/22-23
0x1E–0x1F	Reserved			
0x20	MSCAN Identifier Acceptance Registers (CANIDAR0)	R/W	0x00	22.3.2.16/22-24
0x21	MSCAN Identifier Acceptance Registers (CANIDAR1)	R/W	0x00	22.3.2.16/22-24
0x22–0x23	Reserved			
0x24	MSCAN Identifier Acceptance Registers (CANIDAR2)	R/W	0x00	22.3.2.16/22-24
0x25	MSCAN Identifier Acceptance Registers (CANIDAR3)	R/W	0x00	22.3.2.16/22-24
0x26–0x27	Reserved			
0x28	MSCAN Identifier Mask Registers (CANIDMR0)	R/W	0x00	22.3.2.17/22-25
0x29	MSCAN Identifier Mask Registers (CANIDMR1)	R/W	0x00	22.3.2.17/22-25
0x2A–0x2B	Reserved			
0x2C	MSCAN Identifier Mask Registers (CANIDMR2)	R/W	0x00	22.3.2.17/22-25
0x2D	MSCAN Identifier Mask Registers (CANIDMR3)	R/W	0x00	22.3.2.17/22-25
0x2E–0x2F	Reserved			
0x30	MSCAN Identifier Acceptance Registers (CANIDAR4)	R/W	0x00	22.3.2.17/22-25
0x31	MSCAN Identifier Acceptance Registers (CANIDAR5)	R/W	0x00	22.3.2.17/22-25
0x32–0x33	Reserved			
0x34	MSCAN Identifier Acceptance Registers (CANIDAR6)	R/W	0x00	22.3.2.17/22-25
0x35	MSCAN Identifier Acceptance Registers (CANIDAR7)	R/W	0x00	22.3.2.17/22-25

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x36–0x37	Reserved			
0x38	MSCAN Identifier Mask Registers (CANIDMR4)	R/W	0x00	22.3.2.17/22-25
0x39	MSCAN Identifier Mask Registers (CANIDMR5)	R/W	0x00	22.3.2.17/22-25
0x3A–0x3B	Reserved			
0x3C	MSCAN Identifier Mask Registers (CANIDMR6)	R/W	0x00	22.3.2.17/22-25
0x3D	MSCAN Identifier Mask Registers (CANIDMR7)	R/W	0x00	22.3.2.17/22-25
0x3E–0x3F	Reserved			
0x40–0x5F	**MSCAN Receive Message Buffer (CANRXFG)**			
0x40	RX Identifier Register 0 (IDR0)	R/W	—[3]	22.3.3.1/22-28
0x41	RX Identifier Register 1 (IDR1)	R/W	—[3]	22.3.3.1/22-28
0x42–0x43	Reserved			
0x44	RX Identifier Register 2 (IDR2)	R/W	—[3]	22.3.3.1/22-28
0x45	RX Identifier Register 3 (IDR3)	R/W	—[3]	22.3.3.1/22-28
0x46–0x47	Reserved			
0x48	RX Data Segment Register 0 (DSR0)	R/W	—[3]	22.3.3.2/22-31
0x49	RX Data Segment Register 1 (DSR1)	R/W	—[3]	22.3.3.2/22-31
0x4A–0x4B	Reserved			
0x4C	RX Data Segment Register 2 (DSR2)	R/W	—[3]	22.3.3.2/22-31
0x4D	RX Data Segment Register 3 (DSR3)	R/W	—[3]	22.3.3.2/22-31
0x4E–0x4F	Reserved			
0x50	RX Data Segment Register 4 (DSR4)	R/W	—[3]	22.3.3.2/22-31
0x51	RX Data Segment Register 5 (DSR5)	R/W	—[3]	22.3.3.2/22-31
0x52–0x53	Reserved			
0x54	RX Data Segment Register 6 (DSR6)	R/W	—[3]	22.3.3.2/22-31
0x55	RX Data Segment Register 7 (DSR7)	R/W	—[3]	22.3.3.2/22-31
0x56–0x57	Reserved			
0x58	RX Data Length Register (DLR)	R/W	—[3]	22.3.3.3/22-32
0x59	RX Transmit Buffer Priority Register (TBPR)	R/W	0x00	22.3.3.4/22-32
0x5A–0x5B	Reserved			
0x5C	RX Time Stamp Register High (TSRH)	R	—[3]	22.3.3.5/22-33
0x5D	RX Time Stamp Register Low (TSRL)	R	—[3]	22.3.3.5/22-33
0x5E–0x5F	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x60–0x7F	**MSCAN Transmit Message Buffer (CANTXFG)**			
0x60	TX Identifier Register 0 (IDR0)	R/W	—[3]	22.3.3.1/22-28
0x61	TX Identifier Register 1 (IDR1)	R/W	—[3]	22.3.3.1/22-28
0x62–0x63	Reserved			
0x64	TX Identifier Register 2 (IDR2)	R/W	—[3]	22.3.3.1/22-28
0x65	TX Identifier Register 3 (IDR3)	R/W	—[3]	22.3.3.1/22-28
0x66–0x67	Reserved			
0x68	TX Data Segment Register 0 (DSR0)	R/W	—[3]	22.3.3.2/22-31
0x69	TX Data Segment Register 1 (DSR1)	R/W	—[3]	22.3.3.2/22-31
0x6A–0x6B	Reserved			
0x6C	TX Data Segment Register 2 (DSR2)	R/W	—[3]	22.3.3.2/22-31
0x6D	TX Data Segment Register 3 (DSR3)	R/W	—[3]	22.3.3.2/22-31
0x6E–0x6F	Reserved			
0x70	TX Data Segment Register 4 (DSR4)	R/W	—[3]	22.3.3.2/22-31
0x71	TX Data Segment Register 5 (DSR5)	R/W	—[3]	22.3.3.2/22-31
0x72–0x73	Reserved			
0x74	TX Data Segment Register 6 (DSR6)	R/W	—[3]	22.3.3.2/22-31
0x75	TX Data Segment Register 7 (DSR7)	R/W	—[3]	22.3.3.2/22-31
0x76–0x77	Reserved			
0x78	TX Data Length Register (DLR)	R/W	—[3]	22.3.3.3/22-32
0x79	TX Transmit Buffer Priority Register (TBPR)	R/W	0x00	22.3.3.4/22-32
0x7A–0x7B	Reserved			
0x7C	TX Time Stamp Register High (TSRH)	R	—[3]	22.3.3.5/22-33
0x7D	TX Time Stamp Register Low (TSRL)	R	—[3]	22.3.3.5/22-33
0x7E–0x7F	Reserved			
Fast Ethernet Controller (FEC1) 0xFF40_2800[2] **Chapter 14, “Fast Ethernet Controller (FEC)”**				
0x000	ETH_FEC_ID—FEC Identification Register	R/W	0x0000_0000	14.3.5.1/14-11
0x004	ETH_IEVENT—Interrupt Event Register	R/W	0x0000_0000	14.3.5.2/14-12
0x008	ETH_IMASK—Interrupt Mask Register	R/W	0x0000_0000	14.3.5.3/14-14
0x00C	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x010	ETH_R_DES_ACTIVE—CSR Receive Descriptor Active Register	R/W	0x0000_0000	14.3.5.4/14-15
0x014	ETH_X_DES_ACTIVE—CSR Transmit Descriptor Active Register	R/W	0x0000_0000	14.3.5.5/14-16
0x018–0x023	Reserved			
0x024	ETH_ECNTRL—Ethernet Control register	R/W	0x0000_0000	14.3.5.6/14-17
0x028–0x03C	Reserved			
0x040	ETH_MII_DATA—MII Management Frame Register	R/W	0x0000_0000	14.3.5.7/14-17
0x044	ETH_MII_SPEED—MII Speed Control Register	R/W	0x0000_0000	14.3.5.8/14-19
0x048–0x063	Reserved			
0x064	ETH_MIB_CONTROL—MIB Control Register	R/W	0xC000_0000	14.3.5.9/14-20
0x068–0x083	Reserved			
0x084	ETH_R_CNTRL—Receive Control Register	R/W	0x05EE_0001	14.3.5.10/14-20
0x088	ETH_R_HASH—Receive Hash Register	R		14.3.5.11/14-22
0x08C–0x0C3	Reserved			
0x0C4	ETH_X_CNTRL—Transmit Control Register	R/W	0x0000_0000	14.3.5.12/14-23
0x0C8–0x0E3	Reserved			
0x0E4	ETH_PADDR1—Physical Address Low Register	R/W	0x0000_0000	14.3.5.13/14-24
0x0E8	ETH_PADDR2—Physical Address High Register	R/W	0x0000_8808	14.3.5.14/14-25
0x0EC	ETH_OP_PAUSE—Opcode/Pause Duration Register	R/W	0x0001_UUUU	14.3.5.15/14-25
0x0F0–0x117	Reserved			
0x118	ETH_IADDR1—Descriptor Individual Address 1	R/W	—[3]	14.3.5.16/14-26
0x11C	ETH_IADDR2—Descriptor Individual Address 2	R/W	—[3]	14.3.5.17/14-27
0x120	ETH_GADDR1—Descriptor Group Address 1	R/W	—[3]	14.3.5.18/14-27
0x124	ETH_GADDR2—Descriptor Group Address 2	R/W	—[3]	14.3.5.19/14-28
0x128–0x143	Reserved			
0x144	ETH_X_WMRK—FIFO Transmit FIFO Watermark	R/W	0x0000_0000	14.3.5.20/14-28
0x148	Reserved			
0x14C	ETH_R_BOUND—FIFO Receive Bound Register	R	0x0000_0600	14.3.5.21/14-29
0x150	ETH_R_FSTART—FIFO Receive Start Register	R/W	0x0000_0500	14.3.5.22/14-30
0x154–0x17F	Reserved			
0x180	ETH_R_DES_START—Beginning of Receive Descriptor Ring	R/W	—[3]	14.3.5.23/14-30

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x184	ETH_X_DES_START—Beginning of Transmit Descriptor Ring	R/W	—[3]	14.3.5.24/14-31
0x188	ETH_R_BUFF_SIZE—Receive Buffer Size Register	R/W	—[3]	14.3.5.25/14-32
0x18C–0x1F0	Reserved			
0x1F4	ETH_DMA_CONTROL—DMA Function Control Register	R/W	0xU000_0000	14.3.5.26/14-32
0x1F8–0x1FF	Reserved			
USB ULPI1 0xFF40_3000[2] **Chapter 32, "Universal Serial Bus Interface with On-The-Go"**				
0x000	USB_ID—Identification register	R	0xE241_FA05	33.2.1.1/33-63
0x004	USB_HWGENERAL—General hardware parameters register	R	0x0000_0085	33.2.1.2/33-64
0x008	USB_HWHOST—Host hardware parameters register	R	0x1002_0001	33.2.1.3/33-65
0x00C	USB_HWDEVICE—Device hardware parameters register	R	0x0000_0009	33.2.1.4/33-65
0x010	USB_HWTXBUF—TX buffer hardware parameters register	R	0x1007_0908	33.2.1.4/33-65
0x014	USB_HWRXBUF—RX buffer hardware parameters register	R	0x0000_0808	33.2.1.5/33-66
0x018–0x07F	Reserved			
0x080	USB_GPTIMER0LD—General Purpose Timer load value 0 register	R	0x0000_0000	33.2.2.5/33-72
0x084	USB_GPTIMER0CTRL—General Purpose Timer 0 Control register	R/W	0x0000_0000	33.2.2.5/33-72
0x088	USB_GPTIMER1LD—General Purpose Timer load value 1 register	R	0x0000_0000	33.2.2.5/33-72
0x08C	USB_GPTIMER1CTRL—General Purpose Timer 1 Control register	R/W	0x0000_0000	33.2.2.5/33-72
0x090	USB_SBUSCFG—System Bus Interface Control register	R/W	0x0000_0808	33.2.1.6/33-67
0x094–0x0FF	Reserved			
0x100	USB_CAPLENGTH—Capability register length / HCIVERSION—Host interface version number register	R	0x0100_0040	33.2.2.1/33-68
0x104	USB_HCSPARAMS—Host control structural parameters register	R	0x0001_0011	33.2.2.2/33-69
0x108	USB_HCCPARAMS—Host Control Capability Parameters Register	R	0x0000_0006	33.2.2.3/33-70
0x10C–0x11F	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x120	USB_DCIVERSION—Device Interface Version Number Register	R	0x0000_0000	33.2.2.4/33-71
0x124	USB_DCCPARAMS—Device Controller Parameters Register	R	0x0000_0184	33.2.2.5/33-72
0x128–0x13F	Reserved			
0x140	USB_USBCMD—USB Command Register	R/W	0x0000_0000	33.2.4.1/33-74
0x144	USB_USBSTS—USB Status Register	R/W	0x0000_0000	33.2.4.2/33-77
0x148	USB_USBINTR—USB Interrupt Enable Register	R/W	0x0000_0000	33.2.4.3/33-78
0x14C	USB_FRINDEX—USB Frame Index Register	R/W	0x0000_0000	33.2.4.4/33-82
0x150	Reserved			
0x154	USB_PERIODICLISTBASE—Periodic Frame List Base Address Register	R/W	0x0000_0000	33.2.4.6/33-83
0x154	USB_DEVICEADDR—Device Address Register	R/W	0x0000_0000	33.2.4.7/33-84
0x158	USB_ASYNCLISTADDR—Current Asynchronous List Address Register (Host Mode) (non-EHCI)	R/W	0x0000_0000	33.2.4.8/33-85
0x158	USB_ENDPOINTLISTADDR—Endpoint List Address Register (Device Mode) (non-EHCI)	R/W	0x0000_0000	33.2.4.9/33-86
0x15C	USB_TTCTRL—Host Controller Embedded TT Asynchronous Buffer Status Register	R/W	0x0000_0000	33.2.4.10/33-86
0x160	USB_BURSTSIZE—Master Interface Data Burst Size Register (non-EHCI)	R/W	0x0000_1010	33.2.4.11/33-87
0x164	USB_TXFILLTUNING—Transmit FIFO Tuning Controls Register (non-EHCI)	R/W	0x0000_0000	33.2.4.12/33-88
0x168–0x16F	Reserved			
0x170	USB_ULPIVIEWPORT—ULPI Viewport Register	R/W	0x0800_0000	33.2.4.13/33-89
0x174	Reserved			
0x178	USB_ENDPTNAK—Endpoint NAK Register	R/W	0x0000_0000	33.2.4.14/33-91
0x17C	USB_ENDPTNAKEN—Endpoint NAK Enable Register	R/W	0x0000_0000	33.2.4.15/33-92
0x180	USB_CONFIGFLAG—Configured Flag Register	R	0x0000_0001	33.2.4.16/33-93
0x184	USB_PORTSC0—Port Status/Control Register 0	R/W	0x8C00_0000	33.2.4.17/33-93
0x188	USB_PORTSC1—Port Status/Control Register 1	R/W	0x8C00_0000	33.2.4.17/33-93
0x18C–0x1A0F	Reserved			
0x1A4	USB_OTGSC—On-the-Go Status and Control Register (non-EHCI)	R/W	0x0000_0000	33.2.4.18/33-98
0x1A8	USB_USBMODE—USB Device Mode Register (non-EHCI)	R/W	0x0000_0000	33.2.4.19/33-100

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x1AC	USB_ENDPTSETUPSTAT—Endpoint Setup Status Register (non-EHCI)	R/W	0x0000_0000	33.2.4.20/33-102
0x1B0	USB_ENDPTPRIME—Endpoint Initialization Register (non-EHCI)	R/W	0x0000_0000	33.2.4.21/33-102
0x1B4	USB_ENDPTFLUSH—Endpoint De-initialize Register (non-EHCI)	R/W	0x0000_0000	33.2.4.22/33-103
0x1B8	USB_ENDPTSTATUS—Endpoint Status Register (non-EHCI)	R	0x0000_0000	33.2.4.23/33-104
0x1BC	USB_ENDPTCOMPLETE—Endpoint Complete Register (non-EHCI)	R/W	0x0000_0000	33.2.4.24/33-105
0x01C0	USB_ENDPTCTRL0—Endpoint Control Register 0 (non-EHCI)	R/W	0x0080_0080	33.2.4.25/33-106
0x1C4	USB_ENDPTCTRL1—Endpoint Control Register 1 (non-EHCI)	R/W	0x0000_0000	33.2.4.26/33-107
0x1C8	USB_ENDPTCTRL2—Endpoint Control Register 2 (non-EHCI)	R/W	0x0000_0000	33.2.4.26/33-107
0x1CC	USB_ENDPTCTRL3—Endpoint Control Register 3 (non-EHCI)	R/W	0x0000_0000	33.2.4.26/33-107
0x1D0–0x1FF	Reserved			
0x200	USB_USBGENCTRL — USB General Control Register (non-EHCI)	R/W	0x0000_0000	33.2.4.27/33-109
0x204–0x3FF	Reserved			
USB ULPI2 0xFF40_4000[2] **Chapter 32, "Universal Serial Bus Interface with On-The-Go"**				
0x000	USB_ID—Identification register	R	0xE241_	32.3.1.1/32-22
0x004	USB_HWGENERAL—General hardware parameters register	R	0x0000_0085	32.3.1.2/32-23
0x008	USB_HWHOST—Host hardware parameters register	R	0x1002_0001	32.3.1.3/32-24
0x00C	USB_HWDEVICE—Device hardware parameters register	R	0x0000_0009	32.3.1.4/32-25
0x010	USB_HWTXBUF—TX buffer hardware parameters register	R	0x1007_0908	32.3.1.4/32-25
0x014	USB_HWRXBUF—RX buffer hardware parameters register	R	0x0000_0808	32.3.1.5/32-26
0x018–0x07F	Reserved			
0x080	USB_GPTIMER0LD—General Purpose Timer load value 0 register	R	0x0000_0000	32.3.2.5/32-32

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x084	USB_GPTIMER0CTRL—General Purpose Timer 0 Control register	R/W	0x0000_0000	32.3.2.5/32-32
0x088	USB_GPTIMER1LD—General Purpose Timer load value 1 register	R	0x0000_0000	32.3.2.5/32-32
0x08C	USB_GPTIMER1CTRL—General Purpose Timer 1 Control register	R/W	0x0000_0000	32.3.2.5/32-32
0x090	USB_SBUSCFG—System Bus Interface Control register	R/W	0x0000_0808	32.3.1.6/32-26
0x094–0x0FF	Reserved			
0x100	USB_CAPLENGTH—Capability register length / HCIVERSION—Host interface version number register	R	0x0100_0040	32.3.2.1/32-28
0x104	USB_HCSPARAMS—Host control structural parameters register	R	0x0001_0011	32.3.2.2/32-28
0x108	USB_HCCPARAMS—Host Control Capability Parameters Register	R	0x0000_0006	32.3.2.3/32-29
0x10C–0x11F	Reserved			
0x120	USB_DCIVERSION—Device Interface Version Number Register	R	0x0000_0000	32.3.2.4/32-30
0x124	USB_DCCPARAMS—Device Controller Parameters Register	R	0x0000_0184	32.3.2.5/32-31
0x128–0x13F	Reserved			
0x140	USB_USBCMD—USB Command Register	R/W	0x0000_0000	32.3.4.1/32-34
0x144	USB_USBSTS—USB Status Register	R/W	0x0000_0000	32.3.4.2/32-36
0x148	USB_USBINTR—USB Interrupt Enable Register	R/W	0x0000_0000	32.3.4.3/32-37
0x14C	USB_FRINDEX—USB Frame Index Register	R/W	0x0000_0000	32.3.4.4/32-41
0x150	Reserved			
0x154	USB_PERIODICLISTBASE—Periodic Frame List Base Address Register	R/W	0x0000_0000	32.3.4.6/32-42
0x154	USB_DEVICEADDR—Device Address Register	R/W	0x0000_0000	32.3.4.7/32-43
0x158	USB_ASYNCLISTADDR—Current Asynchronous List Address Register (Host Mode) (non-EHCI)	R/W	0x0000_0000	32.3.4.8/32-44
0x158	USB_ENDPOINTLISTADDR—Endpoint List Address Register (Device Mode) (non-EHCI)	R/W	0x0000_0000	32.3.4.9/32-45
0x15C	USB_TTCTRL—Host Controller Embedded TT Asynchronous Buffer Status Register	R/W	0x0000_0000	32.3.4.10/32-45
0x160	USB_BURSTSIZE—Master Interface Data Burst Size Register (non-EHCI)	R/W	0x0000_1010	32.3.4.11/32-46
0x164	USB_TXFILLTUNING—Transmit FIFO Tuning Controls Register (non-EHCI)	R/W	0x0000_0000	32.3.4.12/32-47

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x168–0x16F	Reserved			
0x170	USB_ULPIVIEWPORT—ULPI Viewport Register	R/W	0x0800_0000	32.3.4.13/32-48
0x174	Reserved			
0x178	USB_ENDPTNAK—Endpoint NAK Register	R/W	0x0000_0000	32.3.4.14/32-50
0x17C	USB_ENDPTNAKEN—Endpoint NAK Enable Register	R/W	0x0000_0000	32.3.4.15/32-51
0x180	USB_CONFIGFLAG—Configured Flag Register	R	0x0000_0001	32.3.4.16/32-52
0x184	USB_PORTSC0—Port Status/Control Register 0	R/W	0x8C00_0000	32.3.4.17/32-52
0x188	USB_PORTSC1—Port Status/Control Register 1	R/W	0x8C00_0000	32.3.4.17/32-52
0x18C–0x1A0F	Reserved			
0x1A4	USB_OTGSC—On-the-Go Status and Control Register (non-EHCI)	R/W	0x0000_0000	32.3.4.18/32-58
0x1A8	USB_USBMODE—USB Device Mode Register (non-EHCI)	R/W	0x0000_0000	32.3.4.19/32-60
0x1AC	USB_ENDPTSETUPSTAT—Endpoint Setup Status Register (non-EHCI)	R/W	0x0000_0000	32.3.4.20/32-61
0x1B0	USB_ENDPTPRIME—Endpoint Initialization Register (non-EHCI)	R/W	0x0000_0000	32.3.4.21/32-62
0x1B4	USB_ENDPTFLUSH—Endpoint De-initialize Register (non-EHCI)	R/W	0x0000_0000	32.3.4.22/32-63
0x1B8	USB_ENDPTSTATUS—Endpoint Status Register (non-EHCI)	R	0x0000_0000	32.3.4.23/32-64
0x1BC	USB_ENDPTCOMPLETE—Endpoint Complete Register (non-EHCI)	R/W	0x0000_0000	32.3.4.24/32-65
0x01C0	USB_ENDPTCTRL0—Endpoint Control Register 0 (non-EHCI)	R/W	0x0080_0080	32.3.4.25/32-66
0x1C4	USB_ENDPTCTRL1—Endpoint Control Register 1 (non-EHCI)	R/W	0x0000_0000	32.3.4.26/32-67
0x1C8	USB_ENDPTCTRL2—Endpoint Control Register 2 (non-EHCI)	R/W	0x0000_0000	32.3.4.26/32-67
0x1CC	USB_ENDPTCTRL3—Endpoint Control Register 3 (non-EHCI)	R/W	0x0000_0000	32.3.4.26/32-67
0x1D0–0x1FF	Reserved			
0x200	USB_USBGENCTRL — USB General Control Register (non-EHCI)	R/W	0x0000_0000	32.3.4.27/32-69
0x204–0x3FF	Reserved			
Fast Ethernet Controller (FEC2) 0xFF40_4800[2] **Chapter 14, "Fast Ethernet Controller (FEC)"**				

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x000	ETH_FEC_ID—FEC Identification Register	R/W	0x0000_0000	14.3.5.1/14-11
0x004	ETH_IEVENT—Interrupt Event Register	R/W	0x0000_0000	14.3.5.2/14-12
0x008	ETH_IMASK—Interrupt Mask Register	R/W	0x0000_0000	14.3.5.3/14-14
0x00C	Reserved			
0x010	ETH_R_DES_ACTIVE—CSR Receive Descriptor Active Register	R/W	0x0000_0000	14.3.5.4/14-15
0x014	ETH_X_DES_ACTIVE—CSR Transmit Descriptor Active Register	R/W	0x0000_0000	14.3.5.5/14-16
0x018–0x023	Reserved			
0x024	ETH_ECNTRL—Ethernet Control register	R/W	0x0000_0000	14.3.5.6/14-17
0x028–0x03C	Reserved			
0x040	ETH_MII_DATA—MII Management Frame Register	R/W	0x0000_0000	14.3.5.7/14-17
0x044	ETH_MII_SPEED—MII Speed Control Register	R/W	0x0000_0000	14.3.5.8/14-19
0x048–0x063	Reserved			
0x064	ETH_MIB_CONTROL—MIB Control Register	R/W	0xC000_0000	14.3.5.9/14-20
0x068–0x083	Reserved			
0x084	ETH_R_CNTRL—Receive Control Register	R/W	0x05EE_0001	14.3.5.10/14-20
0x088	ETH_R_HASH—Receive Hash Register	R		14.3.5.11/14-22
0x08C–0x0C3	Reserved			
0x0C4	ETH_X_CNTRL—Transmit Control Register	R/W	0x0000_0000	14.3.5.12/14-23
0x0C8–0x0E3	Reserved			
0x0E4	ETH_PADDR1—Physical Address Low Register	R/W	0x0000_0000	14.3.5.13/14-24
0x0E8	ETH_PADDR2—Physical Address High Register	R/W	0x0000_8808	14.3.5.14/14-25
0x0EC	ETH_OP_PAUSE—Opcode/Pause Duration Register	R/W	0x0001_UUUU	14.3.5.15/14-25
0x0F0–0x117	Reserved			
0x118	ETH_IADDR1—Descriptor Individual Address 1	R/W	—[3]	14.3.5.16/14-26
0x11C	ETH_IADDR2—Descriptor Individual Address 2	R/W	—[3]	14.3.5.17/14-27
0x120	ETH_GADDR1—Descriptor Group Address 1	R/W	—[3]	14.3.5.18/14-27
0x124	ETH_GADDR2—Descriptor Group Address 2	R/W	—[3]	14.3.5.19/14-28
0x128–0x143	Reserved			
0x144	ETH_X_WMRK—FIFO Transmit FIFO Watermark	R/W	0x0000_0000	14.3.5.20/14-28
0x148	Reserved			
0x14C	ETH_R_BOUND—FIFO Receive Bound Register	R	0x0000_0600	14.3.5.21/14-29

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x150	ETH_R_FSTART—FIFO Receive Start Register	R/W	0x0000_0500	14.3.5.22/14-30
0x154–0x17F	Reserved			
0x180	ETH_R_DES_START—Beginning of Receive Descriptor Ring	R/W	—[3]	14.3.5.23/14-30
0x184	ETH_X_DES_START—Beginning of Transmit Descriptor Ring	R/W	—[3]	14.3.5.24/14-31
0x188	ETH_R_BUFF_SIZE—Receive Buffer Size Register	R/W	—[3]	14.3.5.25/14-32
0x18C–0x1F0	Reserved			
0x1F4	ETH_DMA_CONTROL—DMA Function Control Register	R/W	0xU000_0000	14.3.5.26/14-32
0x1F8–0x1FF	Reserved			
General Purpose Timer (GPT2) 0xFF40_5000[2] **Chapter 15, "General Purpose Timers (GPT)"**				
0x00	GPT0 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x04	GPT0 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x08	GPT0 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x0C	GPT0 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x10	GPT1 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x14	GPT1 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x18	GPT1 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x1C	GPT1 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x20	GPT2 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x24	GPT2 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x28	GPT2 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x2C	GPT2 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x30	GPT3 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x34	GPT3 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x38	GPT3 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x3C	GPT3 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x40	GPT4 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x44	GPT4 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x48	GPT4 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x4C	GPT4 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x50	GPT5 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x54	GPT5 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x58	GPT5 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x5C	GPT5 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x60	GPT6 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x64	GPT6 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x68	GPT6 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x6C	GPT6 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x70	GPT7 Enable and Mode Select Register (GPT_MODE)	R/W	0x0000_0000	15.2.1.1/15-4
0x74	GPT7 Counter Input and up/down counter Output Register (GPT_COUNTER)	R/W	0x0000_0000	15.2.1.2/15-7
0x78	GPT7 PWM Configuration Register (GPT_PWM)	R/W	0x0000_0000	15.2.1.3/15-8
0x7C	GPT7 Status Register (GPT_STATUS)	R	0x0000_0000	15.2.1.4/15-9
0x80–0xFF	Reserved			
Secure Digital Host Controller (SDHC2) 0xFF40_5100[2] **Chapter 28, "Secure Digital Host Controller (SDHC)"**				
0x00	SDHC_STR_STP_CLK—SDHC Clock Control register	R/W	0x0000_0000	28.3.2.1/28-4
0x04	SDHC_STATUS—SDHC Status register	R/W	0x0000_0000	28.3.2.2/28-5
0x08	SDHC_CLK_RATE—SDHC Card Clock Rate register	R/W	0x0000_0008	28.3.2.3/28-9
0x0C	SDHC_CMD_DAT_CONT—SDHC Command and Data Control register	R/W	0x0000_0000	28.3.2.4/28-10
0x10	SDHC_RES_TO—SDHC Response Time-out register	R/W	0x0000_0040	28.3.2.5/28-12
0x14	SDHC_READ_TO—SDHC Read Time-out register	R/W	0x0000_FFFF	28.3.2.6/28-13
0x18	SDHC_BLK_LEN—SDHC Block Length register	R/W	0x0000_0000	28.3.2.7/28-14
0x1C	SDHC_NOB—SDHC Number of Block register	R/W	0x0000_0000	28.3.2.8/28-15
0x20	SDHC_REV_NO—SDHC Revision Number register	R	0x0000_0400	28.3.2.9/28-16
0x24	SDHC_INT_CNTR—SDHC Interrupt Control register	R/W	0x0000_0000	28.3.2.10/28-17
0x28	SDHC_CMD—SDHC Command Number register	R/W	0x0000_0000	28.3.2.11/28-20
0x2C	SDHC_ARG—SDHC Command Argument register	R/W	0x0000_0000	28.3.2.12/28-21
0x30	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x34	SDHC_RES_FIFO—SDHC Command Response FIFO Access register	R	0x0000_0000	28.3.2.13/28-21
0x38	SDHC_BUFFER_ACCESS—SDHC Data Buffer Access register	R/W	0x0000_0000	28.3.2.14/28-22
0x3C–0xFF	Reserved			
Multi-port DRAM controller (MDDRC) 0xFF40_9000[2] **Chapter 11, "DRAM Controller"**				
0x0000	DDR_SYS_CONFIG—DDR System Configuration Register	R/W	0x1000_0000	11.3.2.1/11-5
0x0004	DDR_TIME_CONFIG0—DDR Time Config0 Register	R/W	0x0000_0000	11.3.2.2.1/11-10
0x0008	DDR_TIME_CONFIG1—DDR Time Config1 Register	R/W	0x0000_0000	11.3.2.2.2/11-11
0x000C	DDR_TIME_CONFIG2—DDR Time Config2 Register	R/W	0x0000_0000	11.3.2.2.3/11-11
0x0010	DDR_COMMAND—DRAM Command Register	R/W	0x0000_0000	11.3.2.3/11-15
0x0014	DDR_COMPACT_COMMAND—Compact Command Register	R/W	0x0000_0000	11.3.2.4/11-15
0x0018	SELF_REFRESH_CMD_0—Self-Refresh Command Register 0	R/W	0x0000_0000	11.3.2.5/11-17
0x001C	SELF_REFRESH_CMD_1—Self-Refresh Command Register 1	R/W	0x0000_0000	11.3.2.5/11-17
0x0020	SELF_REFRESH_CMD_2—Self-Refresh Command Register 2	R/W	0x0000_0000	11.3.2.5/11-17
0x0024	SELF_REFRESH_CMD_3—Self-Refresh Command Register 3	R/W	0x0000_0000	11.3.2.5/11-17
0x0028	SELF_REFRESH_CMD_4—Self-Refresh Command Register 4	R/W	0x0000_0000	11.3.2.5/11-17
0x002C	SELF_REFRESH_CMD_5—Self-Refresh Command Register 5	R/W	0x0000_0000	11.3.2.5/11-17
0x0030	SELF_REFRESH_CMD_6—Self-Refresh Command Register 6	R/W	0x0000_0000	11.3.2.5/11-17
0x0034	SELF_REFRESH_CMD_7—Self-Refresh Command Register 7	R/W	0x0000_0000	11.3.2.5/11-17
0x0038	DQS_CONFIG_OFFSET_COUNT—DQS Config Offset Count Register	R/W	0x0000_0000	11.3.2.6/11-18
0x003C	DQS_CONFIG_OFFSET_TIME—DQS Config Offset Time Register	R/W	0x0000_0000	11.3.2.7/11-18
0x0040	DQS_DELAY_STATUS—DQS Delay Status Register	R	0x0000_0000	11.3.2.8/11-19
0x0044–0x005F	Reserved			
0x0060	EXTRA_ATTRIB—DDR Extra Attributes Register	R/W	0x0000_0000	11.3.2.9/11-19

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x0064–0x0FFF	Reserved			
I/O Control 0xFF40_A000[2] Chapter 20, "I/O Control"				
0x000	IO_CTL_MEM—MEM pad control register	R/W	0x00	20.2.2.1/20-6
0x001	IO_CTL_GBOBE—Global Output Enable Control Register	R/W	0x00	20.2.2.2/20-7
0x002–0x003	Reserved			
0x004	IO_CTL_LPC_CLK—LPC_CLK pad control register (STD_PU)	R/W	0x03	20.2.2.3/20-7
0x005	IO_CTL_LPC_OE_B— PC_OE_B pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x006	IO_CTL_LPC_RWB—LPC_RWB pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x007	IO_CTL_LPC_CS0_B—LPC_CS0_B pad control register (STD_PU)	R/W	0x1B	20.2.2.3.1/20-8
0x008	IO_CTL_LPC_ACK_B—LPC_ACK_B pad control register (STD_PU)	R/W	0x1B	20.2.2.3.1/20-8
0x009	IO_CTL_LPC_AX03—LPC_AX03 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x00A	IO_CTL_EMB_AX02—EMB_AX02 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x00B	IO_CTL_EMB_AX01—EMB_AX01 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x00C	IO_CTL_EMB_AX00—EMB_AX00 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x00D	IO_CTL_EMB_AD31—EMB_AD31 pad control register (STD_PU_ST)	R/W	0x03	20.2.2.3.2/20-8
0x00E	IO_CTL_EMB_AD30—EMB_AD30 pad control register (STD_PU_ST)	R/W	0x03	20.2.2.3.2/20-8
0x00F	IO_CTL_EMB_AD29—EMB_AD29 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x010	IO_CTL_EMB_AD28—EMB_AD28 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x011	IO_CTL_EMB_AD27—EMB_AD27 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x012	IO_CTL_EMB_AD26—EMB_AD26 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x013	IO_CTL_EMB_AD25—EMB_AD25 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x014	IO_CTL_EMB_AD24—EMB_AD24 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x015	IO_CTL_EMB_AD23—EMB_AD23 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x016	IO_CTL_EMB_AD22—EMB_AD22 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x017	IO_CTL_EMB_AD21—EMB_AD21 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x018	IO_CTL_EMB_AD20—EMB_AD20 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x019	IO_CTL_EMB_AD19—EMB_AD19 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x01A	IO_CTL_EMB_AD18—EMB_AD18 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x01B	IO_CTL_EMB_AD17—EMB_AD17 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x01C	IO_CTL_EMB_AD16—EMB_AD16 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x01D	IO_CTL_EMB_AD15—EMB_AD15 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x01E	IO_CTL_EMB_AD14—EMB_AD14 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x01R	IO_CTL_EMB_AD13—EMB_AD13 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x020	IO_CTL_EMB_AD12—EMB_AD12 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x021	IO_CTL_EMB_AD11—EMB_AD11 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x022	IO_CTL_EMB_AD10—EMB_AD10 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x023	IO_CTL_EMB_AD09—EMB_AD09 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x024	IO_CTL_EMB_AD08—EMB_AD08 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x025	IO_CTL_EMB_AD07—EMB_AD07 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x026	IO_CTL_EMB_AD06—EMB_AD06 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x027	IO_CTL_EMB_AD05—EMB_AD05 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x028	IO_CTL_EMB_AD04—EMB_AD04 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x029	IO_CTL_EMB_AD03—EMB_AD03 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x02A	IO_CTL_EMB_AD02—EMB_AD02 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x02B	IO_CTL_EMB_AD01—EMB_AD01 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x02C	IO_CTL_EMB_AD00—EMB_AD00 pad control register (STD_PU)	R/W	0x03	20.2.2.3.1/20-8
0x02D	IO_CTL_NFC_CE0_B—NFC_CE0_B pad control register (STD_PU)	R/W	0x1B	20.2.2.3.1/20-8
0x02E	IO_CTL_NFC_RB—NFC_RB pad control register (STD_PU_ST)	R/W	0x07	20.2.2.3.2/20-8
0x02F	IO_CTL_DIU_CLK—DIU_CLK pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x030	IO_CTL_DIU_DE—DIU_DE pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x031	IO_CTL_DIU_HSYNC—DIU_HSYNC pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x032	IO_CTL_DIU_VSYNC—DIU_VSYNC pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x033	IO_CTL_DIU_LD00—DIU_LD00 pad control register (STD_PU_ST)	R/W	0x1C	20.2.2.3.2/20-8
0x034	IO_CTL_DIU_LD01—DIU_LD01 pad control register (STD_PU)	R/W	0x18	20.2.2.3.1/20-8
0x035	IO_CTL_DIU_LD02—DIU_LD02 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x036	IO_CTL_DIU_LD03—DIU_LD03 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x037	IO_CTL_DIU_LD04—DIU_LD04 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x038	IO_CTL_DIU_LD05—DIU_LD05 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x039	IO_CTL_DIU_LD06—DIU_LD06 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x03A	IO_CTL_DIU_LD07—DIU_LD07 pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-8
0x03B	IO_CTL_DIU_LD08—DIU_LD08 pad control register (STD_PU_ST)	R/W	0x1C	20.2.2.3.2/20-8

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x03C	IO_CTL_DIU_LD09—DIU_LD09 pad control register (STD_PU)	R/W	0x18	20.2.2.3.1/20-8
0x03D	IO_CTL_DIU_LD10—DIU_LD10 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x03E	IO_CTL_DIU_LD11—DIU_LD11 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x03F	IO_CTL_DIU_LD12—DIU_LD12 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x040	IO_CTL_DIU_LD13—DIU_LD13 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x041	IO_CTL_DIU_LD14—DIU_LD14 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x042	IO_CTL_DIU_LD15—DIU_LD15 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x043	IO_CTL_DIU_LD16—DIU_LD16 pad control register (STD_PU_ST)	R/W	0x18	20.2.2.3.2/20-8
0x044	IO_CTL_DIU_LD17—DIU_LD17 pad control register (STD_PU_ST)	R/W	0x18	20.2.2.3.2/20-8
0x045	IO_CTL_DIU_LD18—DIU_LD18 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x046	IO_CTL_DIU_LD19—DIU_LD19 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x047	IO_CTL_DIU_LD20—DIU_LD20 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x048	IO_CTL_DIU_LD21—DIU_LD21 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x049	IO_CTL_DIU_LD22—DIU_LD22 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x04A	IO_CTL_DIU_LD23—DIU_LD23 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x04B	IO_CTL_CAN4_RX—I2C2_SCL pad control register (STD_PU_ST)	R/W	0x1C	20.2.2.3.2/20-8
0x04C	IO_CTL_CAN4_TX—I2C2_SDA pad control register (STD_PU_ST)	R/W	0x1C	20.2.2.3.2/20-8
0x04D	IO_CTL_CAN1_TX—CAN1_TX pad control register (STD_PU_ST)	R/W	0x18	20.2.2.3.2/20-8
0x04E	IO_CTL_CAN2_TX—CAN2_TX pad control register (STD_PU_ST)	R/W	0x18	20.2.2.3.2/20-8
0x04F	IO_CTL_I2C1_SCL—I2C1_SCL pad control register (STD_PU_ST)	R/W	0x1C	20.2.2.3.2/20-8

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x050	IO_CTL_I2C1_SDA—I2C1_SDA pad control register (STD_PU_ST)	R/W	0x1C	20.2.2.3.2/20-8
0x051	IO_CTL_FEC1_TXD_2—FEC1_TXD_2 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x052	IO_CTL_FEC1_TXD_3—FEC1_TXD_3 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x053	IO_CTL_FEC1_RXD_2—FEC1_RXD_2 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x054	IO_CTL_FEC1_RXD_3—FEC1_RXD_3 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x055	IO_CTL_FEC1_CRS—FEC1_CRS pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x056	IO_CTL_FEC1_TX_ER—FEC1_TX_ER pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x057	IO_CTL_FEC1_RXD_1—FEC1_RXD_1 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x058	IO_CTL_FEC1_TXD_1—FEC1_TXD_1 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x059	IO_CTL_FEC1_MDC—FEC1_MDC pad control register (STD_PU)	R/W	0x04	20.2.2.3.1/20-8
0x05A	IO_CTL_FEC1_RX_ER—FEC1_RX_ER pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x05B	IO_CTL_FEC1_MDIO—FEC1_MDIO pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-8
0x05C	IO_CTL_FEC1_RXD_0—FEC1_RXD_0 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x05D	IO_CTL_FEC1_TXD_0—FEC1_TXD_0 pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-8
0x05E	IO_CTL_FEC1_TX_CLK—FEC1_TX_CLK pad control register (STD_PU_ST)	R/W	0x04	20.2.2.3.2/20-8
0x05F	IO_CTL_FEC1_RX_CLK—FEC1_RX_CLK pad control register (STD_PU_ST)	R/W	0x04	20.2.2.3.2/20-8
0x060	IO_CTL_FEC1_RX_DV—FEC1_RX_DV pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-8
0x061	IO_CTL_FEC1_TX_EN—FEC1_TX_EN pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x062	IO_CTL_FEC1_COL—FEC1_COL pad control register (STD_PU_ST)	R/W	0x04	20.2.2.3.2/20-8
0x063	IO_CTL_USB1_DATA0—USB1_DATA0 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x064	IO_CTL_USB1_DATA1—USB1_DATA1 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x065	IO_CTL_USB1_DATA2—USB1_DATA2 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x066	IO_CTL_USB1_DATA3—USB1_DATA3 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x067	IO_CTL_USB1_DATA4—USB1_DATA4 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x068	IO_CTL_USB1_DATA5—USB1_DATA5 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x069	IO_CTL_USB1_DATA6—USB1_DATA6 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x06A	IO_CTL_USB1_DATA7—USB1_DATA7 pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-8
0x06B	IO_CTL_USB1_STOP—USB1_STOP pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-8
0x06C	IO_CTL_USB1_CLK—USB1_CLK pad control register (STD_PU_ST)	R/W	0x04	20.2.2.3.2/20-8
0x06D	IO_CTL_USB1_NEXT—USB1_NEXT pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x06E	IO_CTL_USB1_DIR—USB1_DIR pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-8
0x06F	IO_CTL_SDHC1_CLK—SDHC1_CLK pad control register (STD_PU)	R/W	0x18	20.2.2.3.1/20-8
0x070	IO_CTL_SDHC1_CMD—SDHC1_CMD pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x071	IO_CTL_SDHC1_D0—SDHC1_D0 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x072	IO_CTL_SDHC1_D1—SDHC1_D1 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x073	IO_CTL_SDHC1_D2—SDHC1_D2 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x074	IO_CTL_SDHC1_D3—SDHC1_D3 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x075	IO_CTL_PSC_MCLK_IN—PSC_MCLK_IN pad control register (STD_PU_ST)	R/W	0x04	20.2.2.3.2/20-8
0x076	IO_CTL_PSC0_0—PSC0_0 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x077	IO_CTL_PSC0_1—PSC0_1 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x078	IO_CTL_PSC0_2—PSC0_2 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x079	IO_CTL_PSC0_3—PSC0_3 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x07A	IO_CTL_PSC0_4—PSC0_4 pad control register (STD_PU)	R/W	0x18	20.2.2.3.1/20-8
0x07B	IO_CTL_PSC1_0—PSC1_0 pad control register (STD_PU)	R/W	0x18	20.2.2.3.1/20-8
0x07C	IO_CTL_PSC1_1—PSC1_1 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x07D	IO_CTL_PSC1_2—PSC1_2 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x07E	IO_CTL_PSC1_3—PSC1_3 pad control register (STD_PU)	R/W	0x00	20.2.2.3.1/20-8
0x07F	IO_CTL_PSC1_4—PSC1_4 pad control register (STD_PU)	R/W	0x18	20.2.2.3.1/20-8
0x080	IO_CTL_J1850_TX—J1850_TX pad control register (STD_PU_ST)	R/W	0x00	20.2.2.3.2/20-8
0x081	IO_CTL_J1850_RX—J1850_RX pad control register (STD_PU_ST)	R/W	0x18	20.2.2.3.2/20-8
0x082–0xFFF	Reserved			
IC Identification Module (IIM) 0xFF40_B000[2] **Chapter 17, "IIM/Fusebox"**				
0x000	IIM_STAT—Status register	R/W	0x0000_0000	17.3.2.1/17-2
0x004	IIM_STATM—Status IRQ Mask register	R/W	0x0000_0000	17.3.2.2/17-3
0x008	IIM_ERR—Module Errors register	R/W	0x0000_000U	17.3.2.3/17-4
0x00C	IIM_EMASK—Error IRQ Mask register	R/W	0x0000_0000	17.3.2.4/17-5
0x010	IIM_FCTL—Fuse Control register	R/W	0x0000_0030	17.3.2.5/17-6
0x014	IIM_UA—Upper Address register	R/W	0x0000_0000	17.3.2.6/17-7
0x018	IIM_LA—Lower Address register	R/W	0x0000_0000	17.3.2.7/17-8
0x01C	IIM_SDAT—Explicit Sense Data register	R	0x0000_0000	17.3.2.8/17-9
0x020–0x027	Reserved			
0x028	IIM_PREG_P—Program Protection register	R/W	0x0000_0000	17.3.2.9/17-9
0x02C–0x03B	Reserved			
0x03C	IIM_DIVIDE—Divide Factor register	R/W	0x0000_0042	17.3.2.10/17-10
0x040–0xBFF	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0xC00	IIM_FBAC1—Fuse Bank 1 Protection Register	R/W	0x0000_00UU	17.3.2.11/17-11
0xC04–0xC7F	IIM_FB1W1—Fuse Bank 1 Data (available for user)	R/W	0x0000_00UU	17.3.2.12/17-12
0xC80–0xFFF	Reserved			
LocalPlus controller (LPC) 0xFF41_0100[2] **Chapter 21, "LocalPlus Bus Controller (LPC)"**				
0x000	Chip Select 0/Boot Configuration Register (LPC_CS0BOOTC)	R/W	0x0020_UUU1	21.2.1.1.1/21-4
0x004	Chip Select 1 Configuration Register (LPC_CS1C)	R/W	0x0000_0000	21.2.1.1.2/21-7
0x008	Chip Select 2 Configuration Register (LPC_CS2C)	R/W	0x0000_0000	21.2.1.1.2/21-7
0x00C	Chip Select 3 Configuration Register (LPC_CS3C)	R/W	0x0000_0000	21.2.1.1.2/21-7
0x010	Chip Select 4 Configuration Register (LPC_CS4C)	R/W	0x0000_0000	21.2.1.1.2/21-7
0x014	Chip Select 5 Configuration Register (LPC_CS5C)	R/W	0x0000_0000	21.2.1.1.2/21-7
0x018	Chip Select 6 Configuration Register (LPC_CS6C)	R/W	0x0000_0000	21.2.1.1.2/21-7
0x01C	Chip Select 7 Configuration Register (LPC_CS7C)	R/W	0x0000_0000	21.2.1.1.2/21-7
0x020	Chip Select Control Register (LPC_CSC)	R/W	0x0000_0000	21.2.1.1.3/21-9
0x024	Chip Select Status Register (LPC_CSS)	R/W	0x0000_0000	21.2.1.1.4/21-9
0x028	Chip Select Burst Control (LPC_CSBC)	R/W	0x0000_0000	21.2.1.1.5/21-10
0x02C	Chip Select Deadcycle Control Register (LPC_CSDC)	R/W	0x3333_3333	21.2.1.1.6/21-11
0x030	Chip Select Holdcycle Control Register (LPC_CSHC)	R/W	0x3333_3333	21.2.1.1.7/21-12
0x034	Address Latch Timing Register (LPC_ALTR)	R/W	0x0000_0000	21.2.1.1.8/21-12
0x038	TSIZ and TS Enable Register (LPC_TTE)	R/W	0x0000_0000	21.2.1.1.9/21-13
0x03C–0x0FF	Reserved			
0x100	SCLPC Packet Size Register (LPC_SCLPC_PS)	R/W	0x0000_0000	21.2.1.2.1/21-14
0x104	SCLPC Start Address Register (LPC_SCLPC_SA)	R/W	0x0000_0000	21.2.1.2.2/21-15
0x108	SCLPC Control Register (LPC_SCLPC_C)	R/W	0x0001_0000	21.2.1.2.3/21-15
0x10C	SCLPC Enable Register (LPC_SCLPC_E)	R/W	0x0000_0000	21.2.1.2.4/21-16
0x110	Reserved			
0x114	SCLPC Status Register (LPC_SCLPC_S)	R/W	0x0000_0000	21.2.1.2.5/21-17
0x118	SCLPC Bytes Done Register (LPC_SCLPC_BD)	R/W	0x0000_0000	21.2.1.2.6/21-17
0x11C	EMB Share Counter Register (LPC_EMB_SC)	R/W	0x0010_0000	21.2.1.2.7/21-18
0x120	EMB Pause Control Register (LPC_EMB_PC)	R/W	0x0000_0000	21.2.1.2.8/21-18
0x124–0x13F	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x140	LPC RX/TX FIFO Data Word Register (LPC_SCLPC_FIFOD)	R/W	—[3]	21.2.1.3.1/21-19
0x144	LPC RX/TX FIFO Status Register (LPC_SCLPC_FIFOS)	R/W	0x0001_0000	21.2.1.3.2/21-20
0x148	LPC RX/TX FIFO Control Register (LPC_SCLPC_FIFOC)	R/W	0x0100_0000	21.2.1.3.3/21-21
0x14C	LPC RX/TX FIFO Alarm Register (LPC_SCLPC_FIFOA)	R/W	0x0000_0000	21.2.1.3.4/21-21
0x150–0x1FF	Reserved			
Programmable Serial Controller 0 (PSC0) 0xFF41_1000[2] **Chapter 25, "Programmable Serial Controller (PSC)"**				
0x00	Mode Register 1 (MR1)	R/W	—[3]	25.4.1.1/25-6
0x01–0x03	Reserved			
0x04	Mode Register 2 (MR2)	R/W	0x00	25.4.1.2/25-8
0x05–0x07	Reserved			
0x08	Status Register (SR)	R	0x0000	25.4.1.3/25-9
0x0A–0x0B	Reserved			
0x0C	Clock Select Register (CSR)	R/W	0x00	25.4.1.4/25-11
0x10	Command Register (CR)	W	0x00	25.4.1.5/25-12
0x14	Rx Buffer Register (RB)	R	0x0000_0000	25.4.1.6/25-14
0x14	Tx Buffer Register (TB)	W	0x0000_0000	25.4.1.7/25-15
0x18	Input Port Change Register (IPCR)	R	0x00	25.4.1.8/25-17
0x1C	Auxiliary Control Register (ACR)	R/W	0x00	25.4.1.9/25-18
0x20	Interrupt Status Register (ISR)	R	0x0000_0000	25.4.1.10/25-19
0x24	Interrupt Mask Register (IMR)	R/W	0x0000_0000	25.4.1.11/25-20
0x28	Counter Timer Upper Register (CTUR)	R/W	0x00	25.4.1.12/25-22
0x2C	Counter Timer Lower Register (CTLR)	R/W	0x00	25.4.1.13/25-22
0x30	Codec Clock Register (CCR)	R/W	0x0000_0000	25.4.1.14/25-23
0x34	AC97 Slots Register (AC97Slots)	R/W	0x0000_0000	25.4.1.15/25-25
0x38	AC97 Command Register (AC97CMD)	R/W	0x0000_0000	25.4.1.16/25-25
0x3C	AC97 Status Data Register (AC97Data)	R	0x0000_0000	25.4.1.17/25-26
0x40	Reserved			
0x44	Input Port Register (IP)	R	0xU0	25.4.1.18/25-27
0x48	Output Port 1 Bit Set (OP1)	R/W	0x00	25.4.1.19/25-28
0x4C	Output Port 0 Bit Set (OP0)	R/W	0x00	25.4.1.20/25-28

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x50	Serial Interface Control Register (SICR)	R/W	0x0000_0000	25.4.1.21/25-29
0x54–0xFF	FIFOC registers. See Chapter 26, “PSC Centralized FIFO Controller (FIFOC).”			
Programmable Serial Controller 1 (PSC1) 0xFF41_1100[2] Chapter 25, “Programmable Serial Controller (PSC)”				
0x00	Mode Register 1 (MR1)	R/W	—[3]	25.4.1.1/25-6
0x01–0x03	Reserved			
0x04	Mode Register 2 (MR2)	R/W	0x00	25.4.1.2/25-8
0x05–0x07	Reserved			
0x08	Status Register (SR)	R	0x0000	25.4.1.3/25-9
0x0A–0x0B	Reserved			
0x0C	Clock Select Register (CSR)	R/W	0x00	25.4.1.4/25-11
0x10	Command Register (CR)	W	0x00	25.4.1.5/25-12
0x14	Rx Buffer Register (RB)	R	0x0000_0000	25.4.1.6/25-14
0x14	Tx Buffer Register (TB)	W	0x0000_0000	25.4.1.7/25-15
0x18	Input Port Change Register (IPCR)	R	0x00	25.4.1.8/25-17
0x1C	Auxiliary Control Register (ACR)	R/W	0x00	25.4.1.9/25-18
0x20	Interrupt Status Register (ISR)	R	0x0000_0000	25.4.1.10/25-19
0x24	Interrupt Mask Register (IMR)	R/W	0x0000_0000	25.4.1.11/25-20
0x28	Counter Timer Upper Register (CTUR)	R/W	0x00	25.4.1.12/25-22
0x2C	Counter Timer Lower Register (CTLR)	R/W	0x00	25.4.1.13/25-22
0x30	Codec Clock Register (CCR)	R/W	0x0000_0000	25.4.1.14/25-23
0x34	AC97 Slots Register (AC97Slots)	R/W	0x0000_0000	25.4.1.15/25-25
0x38	AC97 Command Register (AC97CMD)	R/W	0x0000_0000	25.4.1.16/25-25
0x3C	AC97 Status Data Register (AC97Data)	R	0x0000_0000	25.4.1.17/25-26
0x40	Reserved			
0x44	Input Port Register (IP)	R	0xU0	25.4.1.18/25-27
0x48	Output Port 1 Bit Set (OP1)	R/W	0x00	25.4.1.19/25-28
0x4C	Output Port 0 Bit Set (OP0)	R/W	0x00	25.4.1.20/25-28
0x50	Serial Interface Control Register (SICR)	R/W	0x0000_0000	25.4.1.21/25-29
0x54–0xFF	FIFOC registers. See Chapter 26, “PSC Centralized FIFO Controller (FIFOC).”			
Programmable Serial Controller 2 (PSC2) 0xFF41_1200[2] Chapter 25, “Programmable Serial Controller (PSC)”				

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x00	Mode Register 1 (MR1)	R/W	—[3]	25.4.1.1/25-6
0x01–0x03	Reserved			
0x04	Mode Register 2 (MR2)	R/W	0x00	25.4.1.2/25-8
0x05–0x07	Reserved			
0x08	Status Register (SR)	R	0x0000	25.4.1.3/25-9
0x0A–0x0B	Reserved			
0x0C	Clock Select Register (CSR)	R/W	0x00	25.4.1.4/25-11
0x10	Command Register (CR)	W	0x00	25.4.1.5/25-12
0x14	Rx Buffer Register (RB)	R	0x0000_0000	25.4.1.6/25-14
0x14	Tx Buffer Register (TB)	W	0x0000_0000	25.4.1.7/25-15
0x18	Input Port Change Register (IPCR)	R	0x00	25.4.1.8/25-17
0x1C	Auxiliary Control Register (ACR)	R/W	0x00	25.4.1.9/25-18
0x20	Interrupt Status Register (ISR)	R	0x0000_0000	25.4.1.10/25-19
0x24	Interrupt Mask Register (IMR)	R/W	0x0000_0000	25.4.1.11/25-20
0x28	Counter Timer Upper Register (CTUR)	R/W	0x00	25.4.1.12/25-22
0x2C	Counter Timer Lower Register (CTLR)	R/W	0x00	25.4.1.13/25-22
0x30	Codec Clock Register (CCR)	R/W	0x0000_0000	25.4.1.14/25-23
0x34	AC97 Slots Register (AC97Slots)	R/W	0x0000_0000	25.4.1.15/25-25
0x38	AC97 Command Register (AC97CMD)	R/W	0x0000_0000	25.4.1.16/25-25
0x3C	AC97 Status Data Register (AC97Data)	R	0x0000_0000	25.4.1.17/25-26
0x40	Reserved			
0x44	Input Port Register (IP)	R	0xU0	25.4.1.18/25-27
0x48	Output Port 1 Bit Set (OP1)	R/W	0x00	25.4.1.19/25-28
0x4C	Output Port 0 Bit Set (OP0)	R/W	0x00	25.4.1.20/25-28
0x50	Serial Interface Control Register (SICR)	R/W	0x0000_0000	25.4.1.21/25-29
0x54–0xFF	FIFOC registers. See Chapter 26, “PSC Centralized FIFO Controller (FIFOC).”			
Programmable Serial Controller 3 (PSC3) 0xFF41_1300[2] **Chapter 25, “Programmable Serial Controller (PSC)”**				
0x00	Mode Register 1 (MR1)	R/W	—[3]	25.4.1.1/25-6
0x01–0x03	Reserved			
0x04	Mode Register 2 (MR2)	R/W	0x00	25.4.1.2/25-8
0x05–0x07	Reserved			

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x08	Status Register (SR)	R	0x0000	25.4.1.3/25-9
0x0A–0x0B	Reserved			
0x0C	Clock Select Register (CSR)	R/W	0x00	25.4.1.4/25-11
0x10	Command Register (CR)	W	0x00	25.4.1.5/25-12
0x14	Rx Buffer Register (RB)	R	0x0000_0000	25.4.1.6/25-14
0x14	Tx Buffer Register (TB)	W	0x0000_0000	25.4.1.7/25-15
0x18	Input Port Change Register (IPCR)	R	0x00	25.4.1.8/25-17
0x1C	Auxiliary Control Register (ACR)	R/W	0x00	25.4.1.9/25-18
0x20	Interrupt Status Register (ISR)	R	0x0000_0000	25.4.1.10/25-19
0x24	Interrupt Mask Register (IMR)	R/W	0x0000_0000	25.4.1.11/25-20
0x28	Counter Timer Upper Register (CTUR)	R/W	0x00	25.4.1.12/25-22
0x2C	Counter Timer Lower Register (CTLR)	R/W	0x00	25.4.1.13/25-22
0x30	Codec Clock Register (CCR)	R/W	0x0000_0000	25.4.1.14/25-23
0x34	AC97 Slots Register (AC97Slots)	R/W	0x0000_0000	25.4.1.15/25-25
0x38	AC97 Command Register (AC97CMD)	R/W	0x0000_0000	25.4.1.16/25-25
0x3C	AC97 Status Data Register (AC97Data)	R	0x0000_0000	25.4.1.17/25-26
0x40	Reserved			
0x44	Input Port Register (IP)	R	0xU0	25.4.1.18/25-27
0x48	Output Port 1 Bit Set (OP1)	R/W	0x00	25.4.1.19/25-28
0x4C	Output Port 0 Bit Set (OP0)	R/W	0x00	25.4.1.20/25-28
0x50	Serial Interface Control Register (SICR)	R/W	0x0000_0000	25.4.1.21/25-29
0x54–0xFF	FIFOC registers. See Chapter 26, “PSC Centralized FIFO Controller (FIFOC).”			
Programmable Serial Controller 4 (PSC4) 0xFF41_1400[2] Chapter 25, “Programmable Serial Controller (PSC)”				
0x00	Mode Register 1 (MR1)	R/W	—[3]	25.4.1.1/25-6
0x01–0x03	Reserved			
0x04	Mode Register 2 (MR2)	R/W	0x00	25.4.1.2/25-8
0x05–0x07	Reserved			
0x08	Status Register (SR)	R	0x0000	25.4.1.3/25-9
0x0A–0x0B	Reserved			
0x0C	Clock Select Register (CSR)	R/W	0x00	25.4.1.4/25-11
0x10	Command Register (CR)	W	0x00	25.4.1.5/25-12

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x14	Rx Buffer Register (RB)	R	0x0000_0000	25.4.1.6/25-14
0x14	Tx Buffer Register (TB)	W	0x0000_0000	25.4.1.7/25-15
0x18	Input Port Change Register (IPCR)	R	0x00	25.4.1.8/25-17
0x1C	Auxiliary Control Register (ACR)	R/W	0x00	25.4.1.9/25-18
0x20	Interrupt Status Register (ISR)	R	0x0000_0000	25.4.1.10/25-19
0x24	Interrupt Mask Register (IMR)	R/W	0x0000_0000	25.4.1.11/25-20
0x28	Counter Timer Upper Register (CTUR)	R/W	0x00	25.4.1.12/25-22
0x2C	Counter Timer Lower Register (CTLR)	R/W	0x00	25.4.1.13/25-22
0x30	Codec Clock Register (CCR)	R/W	0x0000_0000	25.4.1.14/25-23
0x34	AC97 Slots Register (AC97Slots)	R/W	0x0000_0000	25.4.1.15/25-25
0x38	AC97 Command Register (AC97CMD)	R/W	0x0000_0000	25.4.1.16/25-25
0x3C	AC97 Status Data Register (AC97Data)	R	0x0000_0000	25.4.1.17/25-26
0x40	Reserved			
0x44	Input Port Register (IP)	R	0xU0	25.4.1.18/25-27
0x48	Output Port 1 Bit Set (OP1)	R/W	0x00	25.4.1.19/25-28
0x4C	Output Port 0 Bit Set (OP0)	R/W	0x00	25.4.1.20/25-28
0x50	Serial Interface Control Register (SICR)	R/W	0x0000_0000	25.4.1.21/25-29
0x54–0xFF	FIFOC registers. See Chapter 26, “PSC Centralized FIFO Controller (FIFOC).”			
Programmable Serial Controller 5 (PSC5) 0xFF41_1500[2] Chapter 25, “Programmable Serial Controller (PSC)”				
0x00	Mode Register 1 (MR1)	R/W	—[3]	25.4.1.1/25-6
0x01–0x03	Reserved			
0x04	Mode Register 2 (MR2)	R/W	0x00	25.4.1.2/25-8
0x05–0x07	Reserved			
0x08	Status Register (SR)	R	0x0000	25.4.1.3/25-9
0x0A–0x0B	Reserved			
0x0C	Clock Select Register (CSR)	R/W	0x00	25.4.1.4/25-11
0x10	Command Register (CR)	W	0x00	25.4.1.5/25-12
0x14	Rx Buffer Register (RB)	R	0x0000_0000	25.4.1.6/25-14
0x14	Tx Buffer Register (TB)	W	0x0000_0000	25.4.1.7/25-15
0x18	Input Port Change Register (IPCR)	R	0x00	25.4.1.8/25-17
0x1C	Auxiliary Control Register (ACR)	R/W	0x00	25.4.1.9/25-18

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x20	Interrupt Status Register (ISR)	R	0x0000_0000	25.4.1.10/25-19
0x24	Interrupt Mask Register (IMR)	R/W	0x0000_0000	25.4.1.11/25-20
0x28	Counter Timer Upper Register (CTUR)	R/W	0x00	25.4.1.12/25-22
0x2C	Counter Timer Lower Register (CTLR)	R/W	0x00	25.4.1.13/25-22
0x30	Codec Clock Register (CCR)	R/W	0x0000_0000	25.4.1.14/25-23
0x34	AC97 Slots Register (AC97Slots)	R/W	0x0000_0000	25.4.1.15/25-25
0x38	AC97 Command Register (AC97CMD)	R/W	0x0000_0000	25.4.1.16/25-25
0x3C	AC97 Status Data Register (AC97Data)	R	0x0000_0000	25.4.1.17/25-26
0x40	Reserved			
0x44	Input Port Register (IP)	R	0xU0	25.4.1.18/25-27
0x48	Output Port 1 Bit Set (OP1)	R/W	0x00	25.4.1.19/25-28
0x4C	Output Port 0 Bit Set (OP0)	R/W	0x00	25.4.1.20/25-28
0x50	Serial Interface Control Register (SICR)	R/W	0x0000_0000	25.4.1.21/25-29
0x54–0xFF	FIFOC registers. See Chapter 26, “PSC Centralized FIFO Controller (FIFOC).”			
Programmable Serial Controller 6 (PSC6) 0xFF41_1600[2] Chapter 25, “Programmable Serial Controller (PSC)”				
0x00	Mode Register 1 (MR1)	R/W	—[3]	25.4.1.1/25-6
0x01–0x03	Reserved			
0x04	Mode Register 2 (MR2)	R/W	0x00	25.4.1.2/25-8
0x05–0x07	Reserved			
0x08	Status Register (SR)	R	0x0000	25.4.1.3/25-9
0x0A–0x0B	Reserved			
0x0C	Clock Select Register (CSR)	R/W	0x00	25.4.1.4/25-11
0x10	Command Register (CR)	W	0x00	25.4.1.5/25-12
0x14	Rx Buffer Register (RB)	R	0x0000_0000	25.4.1.6/25-14
0x14	Tx Buffer Register (TB)	W	0x0000_0000	25.4.1.7/25-15
0x18	Input Port Change Register (IPCR)	R	0x00	25.4.1.8/25-17
0x1C	Auxiliary Control Register (ACR)	R/W	0x00	25.4.1.9/25-18
0x20	Interrupt Status Register (ISR)	R	0x0000_0000	25.4.1.10/25-19
0x24	Interrupt Mask Register (IMR)	R/W	0x0000_0000	25.4.1.11/25-20
0x28	Counter Timer Upper Register (CTUR)	R/W	0x00	25.4.1.12/25-22
0x2C	Counter Timer Lower Register (CTLR)	R/W	0x00	25.4.1.13/25-22

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x30	Codec Clock Register (CCR)	R/W	0x0000_0000	25.4.1.14/25-23
0x34	AC97 Slots Register (AC97Slots)	R/W	0x0000_0000	25.4.1.15/25-25
0x38	AC97 Command Register (AC97CMD)	R/W	0x0000_0000	25.4.1.16/25-25
0x3C	AC97 Status Data Register (AC97Data)	R	0x0000_0000	25.4.1.17/25-26
0x40	Reserved			
0x44	Input Port Register (IP)	R	0xU0	25.4.1.18/25-27
0x48	Output Port 1 Bit Set (OP1)	R/W	0x00	25.4.1.19/25-28
0x4C	Output Port 0 Bit Set (OP0)	R/W	0x00	25.4.1.20/25-28
0x50	Serial Interface Control Register (SICR)	R/W	0x0000_0000	25.4.1.21/25-29
0x54–0xFF	FIFOC registers. See Chapter 26, "PSC Centralized FIFO Controller (FIFOC)."			
Programmable Serial Controller 7 (PSC7) 0xFF41_1700[2] **Chapter 25, "Programmable Serial Controller (PSC)"**				
0x00	Mode Register 1 (MR1)	R/W	—[3]	25.4.1.1/25-6
0x01–0x03	Reserved			
0x04	Mode Register 2 (MR2)	R/W	0x00	25.4.1.2/25-8
0x05–0x07	Reserved			
0x08	Status Register (SR)	R	0x0000	25.4.1.3/25-9
0x0A–0x0B	Reserved			
0x0C	Clock Select Register (CSR)	R/W	0x00	25.4.1.4/25-11
0x10	Command Register (CR)	W	0x00	25.4.1.5/25-12
0x14	Rx Buffer Register (RB)	R	0x0000_0000	25.4.1.6/25-14
0x14	Tx Buffer Register (TB)	W	0x0000_0000	25.4.1.7/25-15
0x18	Input Port Change Register (IPCR)	R	0x00	25.4.1.8/25-17
0x1C	Auxiliary Control Register (ACR)	R/W	0x00	25.4.1.9/25-18
0x20	Interrupt Status Register (ISR)	R	0x0000_0000	25.4.1.10/25-19
0x24	Interrupt Mask Register (IMR)	R/W	0x0000_0000	25.4.1.11/25-20
0x28	Counter Timer Upper Register (CTUR)	R/W	0x00	25.4.1.12/25-22
0x2C	Counter Timer Lower Register (CTLR)	R/W	0x00	25.4.1.13/25-22
0x30	Codec Clock Register (CCR)	R/W	0x0000_0000	25.4.1.14/25-23
0x34	AC97 Slots Register (AC97Slots)	R/W	0x0000_0000	25.4.1.15/25-25
0x38	AC97 Command Register (AC97CMD)	R/W	0x0000_0000	25.4.1.16/25-25
0x3C	AC97 Status Data Register (AC97Data)	R	0x0000_0000	25.4.1.17/25-26

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x40	Reserved			
0x44	Input Port Register (IP)	R	0xU0	25.4.1.18/25-27
0x48	Output Port 1 Bit Set (OP1)	R/W	0x00	25.4.1.19/25-28
0x4C	Output Port 0 Bit Set (OP0)	R/W	0x00	25.4.1.20/25-28
0x50	Serial Interface Control Register (SICR)	R/W	0x0000_0000	25.4.1.21/25-29
0x54–0xFF	FIFOC registers. See Chapter 26, “PSC Centralized FIFO Controller (FIFOC).”			
Programmable Serial Controller 8 (PSC8) 0xFF41_1800[2] Chapter 25, “Programmable Serial Controller (PSC)”				
0x00	Mode Register 1 (MR1)	R/W	—[3]	25.4.1.1/25-6
0x01–0x03	Reserved			
0x04	Mode Register 2 (MR2)	R/W	0x00	25.4.1.2/25-8
0x05–0x07	Reserved			
0x08	Status Register (SR)	R	0x0000	25.4.1.3/25-9
0x0A–0x0B	Reserved			
0x0C	Clock Select Register (CSR)	R/W	0x00	25.4.1.4/25-11
0x10	Command Register (CR)	W	0x00	25.4.1.5/25-12
0x14	Rx Buffer Register (RB)	R	0x0000_0000	25.4.1.6/25-14
0x14	Tx Buffer Register (TB)	W	0x0000_0000	25.4.1.7/25-15
0x18	Input Port Change Register (IPCR)	R	0x00	25.4.1.8/25-17
0x1C	Auxiliary Control Register (ACR)	R/W	0x00	25.4.1.9/25-18
0x20	Interrupt Status Register (ISR)	R	0x0000_0000	25.4.1.10/25-19
0x24	Interrupt Mask Register (IMR)	R/W	0x0000_0000	25.4.1.11/25-20
0x28	Counter Timer Upper Register (CTUR)	R/W	0x00	25.4.1.12/25-22
0x2C	Counter Timer Lower Register (CTLR)	R/W	0x00	25.4.1.13/25-22
0x30	Codec Clock Register (CCR)	R/W	0x0000_0000	25.4.1.14/25-23
0x34	AC97 Slots Register (AC97Slots)	R/W	0x0000_0000	25.4.1.15/25-25
0x38	AC97 Command Register (AC97CMD)	R/W	0x0000_0000	25.4.1.16/25-25
0x3C	AC97 Status Data Register (AC97Data)	R	0x0000_0000	25.4.1.17/25-26
0x40	Reserved			
0x44	Input Port Register (IP)	R	0xU0	25.4.1.18/25-27
0x48	Output Port 1 Bit Set (OP1)	R/W	0x00	25.4.1.19/25-28
0x4C	Output Port 0 Bit Set (OP0)	R/W	0x00	25.4.1.20/25-28

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x50	Serial Interface Control Register (SICR)	R/W	0x0000_0000	25.4.1.21/25-29
0x54–0xFF	FIFOC registers. See Chapter 26, "PSC Centralized FIFO Controller (FIFOC)."			
Programmable Serial Controller 9 (PSC9) 0xFF41_1900[2] Chapter 25, "Programmable Serial Controller (PSC)"				
0x00	Mode Register 1 (MR1)	R/W	—[3]	25.4.1.1/25-6
0x01–0x03	Reserved			
0x04	Mode Register 2 (MR2)	R/W	0x00	25.4.1.2/25-8
0x05–0x07	Reserved			
0x08	Status Register (SR)	R	0x0000	25.4.1.3/25-9
0x0A–0x0B	Reserved			
0x0C	Clock Select Register (CSR)	R/W	0x00	25.4.1.4/25-11
0x10	Command Register (CR)	W	0x00	25.4.1.5/25-12
0x14	Rx Buffer Register (RB)	R	0x0000_0000	25.4.1.6/25-14
0x14	Tx Buffer Register (TB)	W	0x0000_0000	25.4.1.7/25-15
0x18	Input Port Change Register (IPCR)	R	0x00	25.4.1.8/25-17
0x1C	Auxiliary Control Register (ACR)	R/W	0x00	25.4.1.9/25-18
0x20	Interrupt Status Register (ISR)	R	0x0000_0000	25.4.1.10/25-19
0x24	Interrupt Mask Register (IMR)	R/W	0x0000_0000	25.4.1.11/25-20
0x28	Counter Timer Upper Register (CTUR)	R/W	0x00	25.4.1.12/25-22
0x2C	Counter Timer Lower Register (CTLR)	R/W	0x00	25.4.1.13/25-22
0x30	Codec Clock Register (CCR)	R/W	0x0000_0000	25.4.1.14/25-23
0x34	AC97 Slots Register (AC97Slots)	R/W	0x0000_0000	25.4.1.15/25-25
0x38	AC97 Command Register (AC97CMD)	R/W	0x0000_0000	25.4.1.16/25-25
0x3C	AC97 Status Data Register (AC97Data)	R	0x0000_0000	25.4.1.17/25-26
0x40	Reserved			
0x44	Input Port Register (IP)	R	0xU0	25.4.1.18/25-27
0x48	Output Port 1 Bit Set (OP1)	R/W	0x00	25.4.1.19/25-28
0x4C	Output Port 0 Bit Set (OP0)	R/W	0x00	25.4.1.20/25-28
0x50	Serial Interface Control Register (SICR)	R/W	0x0000_0000	25.4.1.21/25-29
0x54–0xFF	FIFOC registers. See Chapter 26, "PSC Centralized FIFO Controller (FIFOC)."			
Serial FIFO (SFIFO) for PSC 0 to 9 0xFF41_1F00[2] Chapter 25, "Programmable Serial Controller (PSC)"				

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x00	Mode Register 1 (MR1)	R/W	—[3]	25.4.1.1/25-6
0x01–0x03	Reserved			
0x04	Mode Register 2 (MR2)	R/W	0x00	25.4.1.2/25-8
0x05–0x07	Reserved			
0x08	Status Register (SR)	R	0x0000	25.4.1.3/25-9
0x0A–0x0B	Reserved			
0x0C	Clock Select Register (CSR)	R/W	0x00	25.4.1.4/25-11
0x10	Command Register (CR)	W	0x00	25.4.1.5/25-12
0x14	Rx Buffer Register (RB)	R	0x0000_0000	25.4.1.6/25-14
0x14	Tx Buffer Register (TB)	W	0x0000_0000	25.4.1.7/25-15
0x18	Input Port Change Register (IPCR)	R	0x00	25.4.1.8/25-17
0x1C	Auxiliary Control Register (ACR)	R/W	0x00	25.4.1.9/25-18
0x20	Interrupt Status Register (ISR)	R	0x0000_0000	25.4.1.10/25-19
0x24	Interrupt Mask Register (IMR)	R/W	0x0000_0000	25.4.1.11/25-20
0x28	Counter Timer Upper Register (CTUR)	R/W	0x00	25.4.1.12/25-22
0x2C	Counter Timer Lower Register (CTLR)	R/W	0x00	25.4.1.13/25-22
0x30	Codec Clock Register (CCR)	R/W	0x0000_0000	25.4.1.14/25-23
0x34	AC97 Slots Register (AC97Slots)	R/W	0x0000_0000	25.4.1.15/25-25
0x38	AC97 Command Register (AC97CMD)	R/W	0x0000_0000	25.4.1.16/25-25
0x3C	AC97 Status Data Register (AC97Data)	R	0x0000_0000	25.4.1.17/25-26
0x40	Reserved			
0x44	Input Port Register (IP)	R	0xU0	25.4.1.18/25-27
0x48	Output Port 1 Bit Set (OP1)	R/W	0x00	25.4.1.19/25-28
0x4C	Output Port 0 Bit Set (OP0)	R/W	0x00	25.4.1.20/25-28
0x50	Serial Interface Control Register (SICR)	R/W	0x0000_0000	25.4.1.21/25-29
0x54–0xFF	FIFOC registers. See Chapter 26, “PSC Centralized FIFO Controller (FIFOC).”			
DMA 0xFF41_4000[2] Chapter 9, “Direct Memory Access (DMA)”				
0x0000	DMACR—DMA Control Register	R/W	0x0000_E400	9.2.1.1/9-11
0x0004	DMAES—DMA Error Status	R	0x0000_0000	9.2.1.2/9-12
0x0008	DMAERQH—DMA Enable Request High (Channels 63–32)	R/W	0x0000_0000	9.2.1.3/9-14

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x000C	DMAERQL—DMA Enable Request Low (Channels 31–00)	R/W	0x0000_0000	9.2.1.3/9-14
0x0010	DMAEEIH—DMA Enable Error Interrupt High (Channels 63–32)	R/W	0x0000_0000	9.2.1.4/9-15
0x0014	DMAEEIL—DMA Enable Error Interrupt Low (Channels 31–00)	R/W	0x0000_0000	9.2.1.4/9-15
0x0018	DMASERQ—DMA Set Enable Request	R/W	0x00	9.2.1.5/9-16
0x0019	DMACERQ—DMA Clear Enable Request	R/W	0x00	9.2.1.6/9-17
0x001A	DMASEEI—DMA Set Enable Error Interrupt	R/W	0x00	9.2.1.7/9-17
0x001B	DMACEEI—DMA Clear Enable Error Interrupt	R/W	0x00	9.2.1.8/9-18
0x001C	DMACINT—DMA Clear Interrupt Request	R/W	0x00	9.2.1.9/9-19
0x001D	DMACERR—DMA Clear Error	R/W	0x00	9.2.1.10/9-19
0x001E	DMASSRT—DMA Set Start Bit	R/W	0x00	9.2.1.11/9-20
0x001F	DMACDNE—DMA Clear Done Status Bit	R/W	0x00	9.2.1.12/9-20
0x0020	DMAINTH—DMA Interrupt Request High (Channels 63–32)	R/W	0x0000_0000	9.2.1.13/9-21
0x0024	DMAINTL—DMA Interrupt Request Low (Channels 31–00)	R/W	0x0000_0000	9.2.1.13/9-21
0x0028	DMAERRH—DMA Error High (Channels 63–32)	R/W	0x0000_0000	9.2.1.14/9-22
0x002C	DMAERRL—DMA Error Low (Channels 31–00)	R/W	0x0000_0000	9.2.1.14/9-22
0x0030	DMAHRSH—DMA Hardware Request Status High (Channels 63–32)	R	0x0000_0000	9.2.1.15/9-24
0x0034	DMAHRSL—DMA Hardware Request Status Low (Channels 31–00)	R	0x0000_0000	9.2.1.15/9-24
0x0038–0x00FF	Reserved			
0x0100	DCHPRI0—DMA Channel 0 Priority	R/W	0x00	9.2.1.16/9-25
0x0101	DCHPRI1—DMA Channel 1 Priority	R/W	0x00	9.2.1.16/9-25
0x0102	DCHPRI2—DMA Channel 2 Priority	R/W	0x00	9.2.1.16/9-25
0x0103	DCHPRI3—DMA Channel 3 Priority	R/W	0x00	9.2.1.16/9-25
0x0104	DCHPRI4—DMA Channel 4 Priority	R/W	0x00	9.2.1.16/9-25
0x0105	DCHPRI5—DMA Channel 5 Priority	R/W	0x00	9.2.1.16/9-25
0x0106	DCHPRI6—DMA Channel 6 Priority	R/W	0x00	9.2.1.16/9-25
0x0107	DCHPRI7—DMA Channel 7 Priority	R/W	0x00	9.2.1.16/9-25
0x0108	DCHPRI8—DMA Channel 8 Priority	R/W	0x00	9.2.1.16/9-25
0x0109	DCHPRI9—DMA Channel 9 Priority	R/W	0x00	9.2.1.16/9-25

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x010A	DCHPRI10—DMA Channel 10 Priority	R/W	0x00	9.2.1.16/9-25
0x010B	DCHPRI11—DMA Channel 11 Priority	R/W	0x00	9.2.1.16/9-25
0x010C	DCHPRI12—DMA Channel 12 Priority	R/W	0x00	9.2.1.16/9-25
0x010D	DCHPRI13—DMA Channel 13 Priority	R/W	0x00	9.2.1.16/9-25
0x010E	DCHPRI14—DMA Channel 14 Priority	R/W	0x00	9.2.1.16/9-25
0x010F	DCHPRI15—DMA Channel 15 Priority	R/W	0x00	9.2.1.16/9-25
0x0110	DCHPRI16—DMA Channel 16 Priority	R/W	0x00	9.2.1.16/9-25
0x0111	DCHPRI17—DMA Channel 17 Priority	R/W	0x00	9.2.1.16/9-25
0x0112	DCHPRI18—DMA Channel 18 Priority	R/W	0x00	9.2.1.16/9-25
0x0113	DCHPRI19—DMA Channel 19 Priority	R/W	0x00	9.2.1.16/9-25
0x0114	DCHPRI20—DMA Channel 20 Priority	R/W	0x00	9.2.1.16/9-25
0x0115	DCHPRI21—DMA Channel 21 Priority	R/W	0x00	9.2.1.16/9-25
0x0116	DCHPRI22—DMA Channel 22 Priority	R/W	0x00	9.2.1.16/9-25
0x0117	DCHPRI23—DMA Channel 23 Priority	R/W	0x00	9.2.1.16/9-25
0x0118	DCHPRI24—DMA Channel 24 Priority	R/W	0x00	9.2.1.16/9-25
0x0119	DCHPRI25—DMA Channel 25 Priority	R/W	0x00	9.2.1.16/9-25
0x011A	DCHPRI26—DMA Channel 26 Priority	R/W	0x00	9.2.1.16/9-25
0x011B	DCHPRI27—DMA Channel 27 Priority	R/W	0x00	9.2.1.16/9-25
0x011C	DCHPRI28—DMA Channel 28 Priority	R/W	0x00	9.2.1.16/9-25
0x011D	DCHPRI29—DMA Channel 29 Priority	R/W	0x00	9.2.1.16/9-25
0x011E	DCHPRI30—DMA Channel 30 Priority	R/W	0x00	9.2.1.16/9-25
0x011F	DCHPRI31—DMA Channel 31 Priority	R/W	0x00	9.2.1.16/9-25
0x0120	DCHPRI32—DMA Channel 32 Priority	R/W	0x00	9.2.1.16/9-25
0x0121	DCHPRI33—DMA Channel 33 Priority	R/W	0x00	9.2.1.16/9-25
0x0122	DCHPRI34—DMA Channel 34 Priority	R/W	0x00	9.2.1.16/9-25
0x0123	DCHPRI35—DMA Channel 35 Priority	R/W	0x00	9.2.1.16/9-25
0x0124	DCHPRI36—DMA Channel 36 Priority	R/W	0x00	9.2.1.16/9-25
0x0125	DCHPRI37—DMA Channel 37 Priority	R/W	0x00	9.2.1.16/9-25
0x0126	DCHPRI38—DMA Channel 38 Priority	R/W	0x00	9.2.1.16/9-25
0x0127	DCHPRI39—DMA Channel 39 Priority	R/W	0x00	9.2.1.16/9-25
0x0128	DCHPRI40—DMA Channel 40 Priority	R/W	0x00	9.2.1.16/9-25
0x0129	DCHPRI41—DMA Channel 41 Priority	R/W	0x00	9.2.1.16/9-25

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x012A	DCHPRI42—DMA Channel 42 Priority	R/W	0x00	9.2.1.16/9-25
0x012B	DCHPRI43—DMA Channel 43 Priority	R/W	0x00	9.2.1.16/9-25
0x012C	DCHPRI44—DMA Channel 44 Priority	R/W	0x00	9.2.1.16/9-25
0x012D	DCHPRI45—DMA Channel 45 Priority	R/W	0x00	9.2.1.16/9-25
0x012E	DCHPRI46—DMA Channel 46 Priority	R/W	0x00	9.2.1.16/9-25
0x012F	DCHPRI47—DMA Channel 47 Priority	R/W	0x00	9.2.1.16/9-25
0x0130	DCHPRI48—DMA Channel 48 Priority	R/W	0x00	9.2.1.16/9-25
0x0131	DCHPRI49—DMA Channel 49 Priority	R/W	0x00	9.2.1.16/9-25
0x0132	DCHPRI50—DMA Channel 50 Priority	R/W	0x00	9.2.1.16/9-25
0x0133	DCHPRI51—DMA Channel 51 Priority	R/W	0x00	9.2.1.16/9-25
0x0134	DCHPRI52—DMA Channel 52 Priority	R/W	0x00	9.2.1.16/9-25
0x0135	DCHPRI53—DMA Channel 53 Priority	R/W	0x00	9.2.1.16/9-25
0x0136	DCHPRI54—DMA Channel 54 Priority	R/W	0x00	9.2.1.16/9-25
0x0137	DCHPRI55—DMA Channel 55 Priority	R/W	0x00	9.2.1.16/9-25
0x0138	CHPRI56D—DMA Channel 56 Priority)	R/W	0x00	9.2.1.16/9-25
0x0139	DCHPRI57—DMA Channel 57 Priority	R/W	0x00	9.2.1.16/9-25
0x013A	DCHPRI58—DMA Channel 58 Priority	R/W	0x00	9.2.1.16/9-25
0x013B	DCHPRI59—DMA Channel 59 Priority	R/W	0x00	9.2.1.16/9-25
0x013C	DCHPRI60—DMA Channel 60 Priority	R/W	0x00	9.2.1.16/9-25
0x013D	DCHPRI61—DMA Channel 61 Priority	R/W	0x00	9.2.1.16/9-25
0x013E	DCHPRI62—DMA Channel 62 Priority	R/W	0x00	9.2.1.16/9-25
0x013F	DCHPRI63—DMA Channel 63 Priority	R/W	0x00	9.2.1.16/9-25
0x0140–0x0FFC	Reserved			
0x1000–0x101C	TCD00—Transfer Control Descriptors Channel 00	R/W	—[3]	9.2.1.17/9-25
0x1020–0x103C	TCD01—Transfer Control Descriptors Channel 01	R/W	—[3]	9.2.1.17/9-25
0x1040–0x105C	TCD02—Transfer Control Descriptors Channel 02	R/W	—[3]	9.2.1.17/9-25
0x1060–0x107C	TCD03—Transfer Control Descriptors Channel 03	R/W	—[3]	9.2.1.17/9-25
0x1080–0x109C	TCD04—Transfer Control Descriptors Channel 04	R/W	—[3]	9.2.1.17/9-25
0x10A0–0x10BC	TCD05—Transfer Control Descriptors Channel 05	R/W	—[3]	9.2.1.17/9-25
0x10C0–0x10DC	TCD06—Transfer Control Descriptors Channel 06	R/W	—[3]	9.2.1.17/9-25
0x10E0–0x10FC	TCD07—Transfer Control Descriptors Channel 07	R/W	—[3]	9.2.1.17/9-25
0x1100–0x113C	TCD08—Transfer Control Descriptors Channel 08	R/W	—[3]	9.2.1.17/9-25

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x1120–0x113C	TCD09—Transfer Control Descriptors Channel 09	R/W	—[3]	9.2.1.17/9-25
0x1140–0x115C	TCD10—Transfer Control Descriptors Channel 10	R/W	—[3]	9.2.1.17/9-25
0x1160–0x117C	TCD11—Transfer Control Descriptors Channel 11	R/W	—[3]	9.2.1.17/9-25
0x1180–0x119C	TCD12—Transfer Control Descriptors Channel 12	R/W	—[3]	9.2.1.17/9-25
0x11A0–0x11BC	TCD13—Transfer Control Descriptors Channel 13	R/W	—[3]	9.2.1.17/9-25
0x11C0–0x11DC	TCD14—Transfer Control Descriptors Channel 14	R/W	—[3]	9.2.1.17/9-25
0x11E0–0x11FC	TCD15—Transfer Control Descriptors Channel 15	R/W	—[3]	9.2.1.17/9-25
0x1200–0x121C	TCD16—Transfer Control Descriptors Channel 16	R/W	—[3]	9.2.1.17/9-25
0x1220–0x123C	TCD17—Transfer Control Descriptors Channel 17	R/W	—[3]	9.2.1.17/9-25
0x1240–0x125C	TCD18—Transfer Control Descriptors Channel 18	R/W	—[3]	9.2.1.17/9-25
0x1260–0x127C	TCD19—Transfer Control Descriptors Channel 19	R/W	—[3]	9.2.1.17/9-25
0x1280–0x129C	TCD20—Transfer Control Descriptors Channel 20	R/W	—[3]	9.2.1.17/9-25
0x12A0–0x12BC	TCD21—Transfer Control Descriptors Channel 21	R/W	—[3]	9.2.1.17/9-25
0x12C0–0x12DC	TCD22—Transfer Control Descriptors Channel 22	R/W	—[3]	9.2.1.17/9-25
0x12E0–0x12FC	TCD23—Transfer Control Descriptors Channel 23	R/W	—[3]	9.2.1.17/9-25
0x1300–0x133C	TCD24—Transfer Control Descriptors Channel 24	R/W	—[3]	9.2.1.17/9-25
0x1320–0x133C	TCD25—Transfer Control Descriptors Channel 25	R/W	—[3]	9.2.1.17/9-25
0x1340–0x135C	TCD26—Transfer Control Descriptors Channel 26	R/W	—[3]	9.2.1.17/9-25
0x1360–0x137C	TCD27—Transfer Control Descriptors Channel 27	R/W	—[3]	9.2.1.17/9-25
0x1380–0x139C	TCD28—Transfer Control Descriptors Channel 28	R/W	—[3]	9.2.1.17/9-25
0x13A0–0x13BC	TCD29—Transfer Control Descriptors Channel 29	R/W	—[3]	9.2.1.17/9-25
0x13C0–0x13DC	TCD30—Transfer Control Descriptors Channel 30	R/W	—[3]	9.2.1.17/9-25
0x13E0–0x13FC	TCD31—Transfer Control Descriptors Channel 31	R/W	—[3]	9.2.1.17/9-25
0x1400–0x141C	TCD32—Transfer Control Descriptors Channel 32	R/W	—[3]	9.2.1.17/9-25
0x1420–0x143C	TCD33—Transfer Control Descriptors Channel 33	R/W	—[3]	9.2.1.17/9-25
0x1440–0x145C	TCD34—Transfer Control Descriptors Channel 34	R/W	—[3]	9.2.1.17/9-25
0x1460–0x147C	TCD35—Transfer Control Descriptors Channel 35	R/W	—[3]	9.2.1.17/9-25
0x1480–0x149C	TCD36—Transfer Control Descriptors Channel 36	R/W	—[3]	9.2.1.17/9-25
0x14A0–0x14BC	TCD37—Transfer Control Descriptors Channel 37	R/W	—[3]	9.2.1.17/9-25
0x14C0–0x14DC	TCD38—Transfer Control Descriptors Channel 38	R/W	—[3]	9.2.1.17/9-25
0x15E0–0x14FC	TCD39—Transfer Control Descriptors Channel 39	R/W	—[3]	9.2.1.17/9-25
0x1500–0x153C	TCD40—Transfer Control Descriptors Channel 40	R/W	—[3]	9.2.1.17/9-25

Table A-2. MPC5125 Detailed Register Map (continued)

Address Offset from Module Base	Register	Access	Reset Value[1]	Section/Page
0x1520–0x153C	TCD41—Transfer Control Descriptors Channel 41	R/W	—[3]	9.2.1.17/9-25
0x1540–0x155C	TCD42—Transfer Control Descriptors Channel 42	R/W	—[3]	9.2.1.17/9-25
0x1560–0x157C	TCD43—Transfer Control Descriptors Channel 43	R/W	—[3]	9.2.1.17/9-25
0x1580–0x159C	TCD44—Transfer Control Descriptors Channel 44	R/W	—[3]	9.2.1.17/9-25
0x15A0–0x115BC	TCD45—Transfer Control Descriptors Channel 45	R/W	—[3]	9.2.1.17/9-25
0x15C0–0x15DC	TCD46—Transfer Control Descriptors Channel 46	R/W	—[3]	9.2.1.17/9-25
0x15E0–0x15FC	TCD47—Transfer Control Descriptors Channel 47	R/W	—[3]	9.2.1.17/9-25
0x1600–0x161C	TCD48—Transfer Control Descriptors Channel 48	R/W	—[3]	9.2.1.17/9-25
0x1620–0x163C	TCD49—Transfer Control Descriptors Channel 49	R/W	—[3]	9.2.1.17/9-25
0x1640–0x165C	TCD50—Transfer Control Descriptors Channel 50	R/W	—[3]	9.2.1.17/9-25
0x1660–0x167C	TCD51—Transfer Control Descriptors Channel 51	R/W	—[3]	9.2.1.17/9-25
0x1680–0x169C	TCD52—Transfer Control Descriptors Channel 52	R/W	—[3]	9.2.1.17/9-25
0x16A0–0x16BC	TCD53—Transfer Control Descriptors Channel 53	R/W	—[3]	9.2.1.17/9-25
0x16C0–0x16DC	TCD54—Transfer Control Descriptors Channel 54	R/W	—[3]	9.2.1.17/9-25
0x16E0–0x16FC	TCD55—Transfer Control Descriptors Channel 55	R/W	—[3]	9.2.1.17/9-25
0x1700–0x173C	TCD56—Transfer Control Descriptors Channel 56	R/W	—[3]	9.2.1.17/9-25
0x1720–0x1713C	TCD57—Transfer Control Descriptors Channel 57	R/W	—[3]	9.2.1.17/9-25
0x1740–0x175C	TCD58—Transfer Control Descriptors Channel 58	R/W	—[3]	9.2.1.17/9-25
0x1760–0x177C	TCD59—Transfer Control Descriptors Channel 59	R/W	—[3]	9.2.1.17/9-25
0x1780–0x179C	TCD60—Transfer Control Descriptors Channel 60	R/W	—[3]	9.2.1.17/9-25
0x17A0–0x17BC	TCD61—Transfer Control Descriptors Channel 61	R/W	—[3]	9.2.1.17/9-25
0x117C0–0x171DC	TCD62—Transfer Control Descriptors Channel 62	R/W	—[3]	9.2.1.17/9-25
0x17E0–0x17FC	TCD63—Transfer Control Descriptors Channel 63	R/W	—[3]	9.2.1.17/9-25
0x1800–0x3FFF	Reserved			

1 In this column, the symbol “U” indicates one or more bits in a byte are undefined at reset. See the associated description for more information.

2 Default absolute offset with IMMRBAR at default location of 0xFF40_0000. See Chapter 2, “System Configuration and Memory Map (XLBMEN + Mem Map).”

3 Reset value is indeterminate. See the associated description for more information.

This page is intentionally left blank.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

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)
www.freescale.com/support

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 China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

MPC5125RM
Rev. 3
10/2011