

Zynq-7000 EPP Technical Reference Manual

UG585 (v1.2) August 8, 2012



Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.

© Copyright 2012 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. CPRI is a trademark of Siemens AG. PCI, PCI Express, PCIe, and PCI-X are trademarks of PCI-SIG. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for this document. Change bars indicate the latest revisions.

Date	Version	Revision
04/08/12	1.0	Xilinx initial release.

Date	Version	Revision
06/25/12	1.1	Removed Chapter 30, Board Design (now part of UG933, <i>Zynq-7000 EPP PCB Design and Pin Planning Guide</i>).
08/08/12	1.2	<p>Added information about the 7z010 CLG225 device and references to section 2.4.4 MIO-at-a-Glance Table throughout document. Added section headings 1.1.1 Block Diagram and 1.1.2 Documentation Resources, added sections 1.1.3 Notices and TrustZone Capabilities, and clarified PS MIO I/Os in Chapter 1. Updated Table 2-1. Changed 2.4.2 MIO-EMIO Connections heading to 2.4.2 IOP Interface Connections and clarified first paragraph. Updated Table 2-4. Added section 2.4.8 PS-PL Voltage Level Shifter Enables and Table 2-7, and updated Table 2-13 PS MIO I/Os in Chapter 2. Added note under Branch Prediction and Table 3-4 in Chapter 3. Updated Table 4-1 in Chapter 4. Added section 5.1.7 Read/Write Request Capability in Chapter 5. Updated NAND MIO pin assignments and Table 6-6 in Chapter 6. Updated section 7.2 Functional Description in Chapter 7. Added section heading 10.1.1 Features and added section 10.1.3 Notices in Chapter 10. Updated Parallel (SRAM/NOR) Interface features list and added section 11.1.3 Notices in Chapter 11. Reorganized, clarified, and expanded Chapter 12 to include programming models (added sections 12.1.4 Notices, 12.3 Programming Guide, and 12.5.2 MIO Programming). Added last note in section 13.3.4 Using ADMA in Chapter 13. Added Restrictions in Chapter 14. Clarified first paragraph, added section 15.1.3 Notices, and clarified Figure 15-7 through Figure 15-11 in Chapter 15. Added section 16.1.4 Notices in Chapter 16. Clarified sections 17.2.5 SPI FIFOs, 17.2.6 SPI Clocks, and 17.2.7 SPI EMIO Considerations in Chapter 17. Reorganized, clarified, and expanded Chapter 18 to include programming models (added sections 18.1.4 Notices and 18.5.1 MIO Programming). Reorganized, clarified, and expanded Chapter 19 to include programming models (added sections 19.1.4 Notices, 19.3 Programming Guide, and 19.5.1 MIO Programming). Updated Table 22-2 and Table 22-3 in Chapter 22. Added section 25.12.5 CPU_DIVISOR(ARM_CLK_CTRL[13:8]) in Chapter 25. Updated Table 26-4 in Chapter 26. Clarified section 27.3 I/O Signals in Chapter 27. Added section 28.1.2 Notices in Chapter 28. Clarified Mapping Summary and updated Table 29-1, Table 29-3, and Table 29-5 in Chapter 29. Added section 30.1.3 Notices in Chapter 30. Updated data sheet references in section A.3.1 Zynq-7000 EPP Documents of Appendix A. Updated register database in sections B.3 Module Summary through B.34 USB Controller (usb) in Appendix B.</p>

Table of Contents

Revision History	2
------------------------	---

Chapter 1: Introduction

1.1 Overview	21
1.1.1 Block Diagram	22
1.1.2 Documentation Resources	23
1.1.3 Notices	24
1.2 Processing System (PS) Features and Descriptions	25
1.2.1 Application Processor Unit (APU)	25
1.2.2 Memory Interfaces	26
1.2.3 I/O Peripherals	28
1.3 Programmable Logic Features and Descriptions	32
1.4 Interconnect Features and Description	33
1.4.1 PS Interconnect Based on AXI High Performance Data Path Switches	33
1.4.2 PS-PL Interfaces	34
1.5 System Software	35

Chapter 2: Signals, Interfaces, and Pins

2.1 Introduction	36
2.2 Power Pins	36
2.3 PS I/O Pins	38
2.4 MIO-EMIO	38
2.4.1 I/O Peripheral (IOP) Interface Routing	39
2.4.2 IOP Interface Connections	40
2.4.3 MIO Pin Assignment Considerations	41
2.4.4 MIO-at-a-Glance Table	43
2.4.5 MIO Signal Routing	44
2.4.6 Default Logic Levels	44
2.4.7 MIO Pin Electrical Parameters	45
2.4.8 PS-PL Voltage Level Shifter Enables	46
2.5 PL AXI Interfaces	46
2.6 PL-PS Signals	46
2.6.1 Clocks and Resets	47
2.6.2 Interrupt Signals	47
2.6.3 Event Signals	48
2.6.4 Idle AXI, DDR Urgent/Arb, SRAM Interrupt Signals	48
2.6.5 DMA Req/Ack Signals	49
2.7 PL I/O Pins	49

Chapter 3: Application Processing Unit

3.1 Introduction	51
3.1.1 Basic Functionality	51
3.1.2 System-level View	53
3.2 Cortex A9 Processors	55
3.2.1 Summary	55
3.2.2 Central Processing Unit (CPU)	55
3.2.3 Level 1 Caches	58
3.2.4 Memory Management Unit (MMU)	60
3.2.5 Interfaces	63
3.2.6 NEON	64
3.2.7 Performance Monitoring Unit	65
3.3 Snoop Control Unit (SCU)	65
3.3.1 Summary	65
3.3.2 Address Filtering	66
3.3.3 SCU Master Ports	66
3.4 L2-Cache	66
3.4.1 Summary	66
3.4.2 Exclusive L2-L1 Cache Configuration	70
3.4.3 Cache Replacement Strategy	71
3.4.4 Cache Lockdown	71
3.4.5 Enabling and Disabling the L2 Cache Controller	72
3.4.6 RAM Access Latency Control	73
3.4.7 Store Buffer Operation	73
3.4.8 Optimizations Between Cortex-A9 and L2 Controller	74
3.4.9 Pre-fetching Operation	75
3.4.10 Programming Model	76
3.5 APU Interfaces	77
3.5.1 PL Co-processing Interfaces	77
3.5.2 Interrupt Interface	80
3.6 Support for TrustZone Within the APU	81
3.6.1 CPU Security Transition	81
3.6.2 CP15 Register Access Control	82
3.6.3 MMU Security	83
3.6.4 L-1 Cache Security	83
3.6.5 Security Exception Control	84
3.6.6 CPU Debug TrustZone Access Control	84
3.6.7 SCU Register Access Control	84
3.6.8 TrustZone Support in the L2 Cache	84
3.7 Application Processing Unit (APU) Reset	85
3.7.1 Reset Functionality	85
3.7.2 APU State After Reset	86
3.8 Power Considerations	86
3.8.1 Introduction	86
3.8.2 Standby Mode	87
3.8.3 Dynamic Clock Gating in the L2 Controller	87

Chapter 4: System Addresses

4.1 Address Map	88
4.2 System Bus Masters	90
4.3 SLCR Registers	90
4.4 CPU Private Bus Registers	91
4.5 SMC Memory	92
4.6 PS I/O Peripherals	92
4.7 Miscellaneous PS Registers	93

Chapter 5: Interconnect

5.1 Introduction	94
5.1.1 Features	94
5.1.2 Block Diagram	94
5.1.3 Datapaths	97
5.1.4 Clock Domains	98
5.1.5 Connectivity	100
5.1.6 AXI ID	100
5.1.7 Read/Write Request Capability	101
5.1.8 Register Overview	101
5.2 Quality of Service	102
5.2.1 Basic Arbitration	102
5.2.2 Advanced QoS	102
5.2.3 DDR Port Arbitration	103
5.3 AXI_HP Interfaces	103
5.3.1 Features	103
5.3.2 Block Diagram	104
5.3.3 Functional Description	105
5.3.4 Performance	105
5.3.5 Register Overview	106
5.3.6 Bandwidth Management Features	106
5.3.7 Transaction Types	110
5.3.8 Command Interleaving and Re-Ordering	110
5.3.9 Performance Optimization Summary	111
5.4 AXI_ACP Interface	112
5.5 AXI_GP Interfaces	113
5.5.1 Features	113
5.5.2 Performance	113
5.6 Signals	113
5.6.1 AXI Signals	113
5.6.2 Clocks and Resets	117

Chapter 6: Boot and Configuration

6.1 Introduction	118
6.2 External Startup Requirements	119
6.2.1 Introduction	119
6.2.2 Power Requirements	119
6.2.3 Clock Requirements	119

6.2.4	Reset Requirements	119
6.2.5	Mode Pin Settings	120
6.3	BootROM	121
6.3.1	Introduction and Features	121
6.3.2	BootROM Header	125
6.3.3	Boot Performance	128
6.3.4	Boot Devices	129
6.3.5	BootROM MultiBoot and Boot Partition Search	134
6.3.6	Debug Status	137
6.3.7	Post BootROM State	138
6.4	Device Configuration Interface	140
6.4.1	Block Diagram	141
6.4.2	Features	141
6.4.3	Functional Description	142
6.4.4	Device Configuration Flow	145
6.4.5	PL Configuration	149
6.4.6	Register Overview	151

Chapter 7: Interrupts

7.1	Environment	152
7.1.1	Private, Shared and Software Interrupts	153
7.1.2	Generic Interrupt Controller (GIC)	153
7.1.3	Resets and Clocks	153
7.1.4	Block Diagram	153
7.2	Functional Description	155
7.2.1	Software Generated Interrupts (SGI)	155
7.2.2	CPU Private Peripheral Interrupts (PPI)	155
7.2.3	Shared Peripheral Interrupts (SPI)	156
7.2.4	Wait for Interrupt Event Signal (WFI)	157
7.3	Register Overview	158
7.3.1	Write Protection Lock Down	159
7.4	Programming Model	159
7.4.1	Interrupt Prioritization	159
7.4.2	Interrupt Handling	159
7.4.3	ARM Programming Topics	160
7.4.4	Legacy Interrupts and Security Extensions	160

Chapter 8: Timers

8.1	Introduction	161
8.1.1	System Diagram	162
8.2	CPU Private Timers and Watchdog Timers	162
8.2.1	Clocking	163
8.2.2	Register Overview	163
8.3	Global Timer (GT)	163
8.3.1	Clocking	163
8.3.2	Register Overview	164
8.4	System Watchdog Timer (SWDT)	164
8.4.1	Features	164
8.4.2	Block Diagram	165

8.4.3 Functional Description	165
8.4.4 Register Overview	166
8.4.5 Programming Model	166
8.5 Triple Timer Counters (TTC)	167
8.5.1 Features	167
8.5.2 Block Diagram	167
8.5.3 Functional Description	168
8.5.4 Register Overview	169
8.5.5 Programming Model	170
8.5.6 I/O Signals	171

Chapter 9: DMA Controller

9.1 Introduction	173
9.1.1 Features, Constraints and Limitations	174
9.1.2 System Viewpoint	176
9.1.3 Block Diagram	177
9.2 Functional Description	179
9.2.1 Programming the DMA Controller	181
9.2.2 Memory to Memory Transaction	182
9.2.3 Memory to/from PL Peripheral Transaction	183
9.2.4 Multi-channel Data FIFO (MFIFO)	188
9.2.5 Events and Interrupts	188
9.2.6 Aborts	189
9.2.7 Security Usage	191
9.3 External Signals	193
9.3.1 Peripheral Request Interface	193
9.3.2 AXI Master Interface	194
9.3.3 Reset Initialization Interface	195
9.4 Register Overview	196
9.5 Instruction Set Reference for Manager and Commands	197
9.6 Programming Examples Reference	198
9.6.1 Start a DMA Channel Thread	198
9.6.2 Memory to Memory	199
9.6.3 Memory to/from PL Peripheral	202
9.6.4 Using an Event to Restart DMA Channels	204
9.6.5 Interrupting a Processor	205
9.7 Programming Restrictions	205
9.8 DMAC IP Configuration Options	208

Chapter 10: DDR Memory Controller

10.1 Introduction	209
10.1.1 Features	209
10.1.2 Block Diagram	210
10.1.3 Notices	211
10.1.4 Interconnect	211
10.1.5 DDR Memory Types, Densities, and Data Widths	212
10.1.6 I/O Signals	213
10.2 AXI Memory Port Interface (DDR1)	214
10.2.1 Introduction	214

10.2.2	Block Diagram	215
10.2.3	AXI Feature Support and Limitations	215
10.2.4	TrustZone	216
10.3	DDR Core and Transaction Scheduler (DDRC)	216
10.3.1	Row/Bank/Column Address Mapping	218
10.4	DDRC Arbitration	218
10.4.1	Priority, Aging Counter and Urgent Signals	219
10.4.2	Page-Match	219
10.4.3	Aging Counter	220
10.4.4	Stage 1 – AXI Port arbitration	220
10.4.5	Stage 2 – Read Versus Write	222
10.4.6	High Priority Read Ports	222
10.4.7	Stage 3 – Transaction State	223
10.4.8	Read Priority Management	225
10.4.9	Write Combine	225
10.4.10	Credit Mechanism	226
10.5	Controller PHY (DDRP)	226
10.5.1	Loopback	227
10.6	Initialization and Calibration	227
10.6.1	DDR Clock Initialization	227
10.6.2	DDR IOB Impedance Calibration	228
10.6.3	DDR IOB Configuration	229
10.6.4	DDR Controller Register Programming	231
10.6.5	DRAM Reset and Initialization	231
10.6.6	DRAM Input Impedance (ODT) Calibration	231
10.6.7	DRAM Output Impedance (R_{ON}) Calibration	231
10.6.8	DRAM Training	232
10.6.9	Write Data Eye Adjustment	234
10.6.10	ECC Initialization	234
10.6.11	Alternatives to Automatic DRAM Training	234
10.7	Register Overviews	237
10.7.1	DDRI	237
10.7.2	DDRC	237
10.7.3	DDRP	239
10.8	Error Correction Code (ECC)	240
10.8.1	ECC Error Behavior	240
10.8.2	Data Mask During ECC Mode	240
10.9	Programming Model	241
10.9.1	Changing Clock Frequencies	241
10.9.2	ECC Programming	241
10.9.3	Power Down	242
10.9.4	Self Refresh	243
10.9.5	Deep Power Down	243

Chapter 11: Static Memory Controller

11.1	Introduction	244
11.1.1	Features	245
11.1.2	Block Diagram	246
11.1.3	Notices	247
11.2	Functional Operation	247

11.2.1	Boot Device	247
11.2.2	Clocks	247
11.2.3	Resets	247
11.2.4	Interrupts	248
11.2.5	PL353 Functionality	248
11.2.6	Address Map	248
11.3	I/O Signals	249
11.4	Wiring Diagrams	250
11.5	Register Overview	251
11.6	Programming Model	252

Chapter 12: Quad-SPI Flash Controller

12.1	Introduction	253
12.1.1	Features	253
12.1.2	System Viewpoint	254
12.1.3	Block Diagram	255
12.1.4	Notices	255
12.2	Functional Description	256
12.2.1	Operational Modes	256
12.2.2	I/O Mode	256
12.2.3	Linear Addressing Mode	258
12.3	Programming Guide	261
12.3.1	Configuration	261
12.3.2	Configure Linear Addressing Mode	262
12.3.3	Configure I/O Mode	262
12.3.4	I/O Mode Interrupts	265
12.3.5	Rx/Tx FIFO Response to I/O Command Sequences	265
12.3.6	Register Overview	268
12.4	System Functions	268
12.4.1	Clocks	268
12.4.2	Resets	270
12.5	I/O Interface	270
12.5.1	Wiring Connections	270
12.5.2	MIO Programming	273
12.5.3	MIO Signals	275

Chapter 13: SD/SDIO Peripheral Controller

13.1	Introduction	276
13.1.1	Key Features	277
13.1.2	System Viewpoint	278
13.2	Functional Description	278
13.2.1	AHB Interface and Interrupt Controller	278
13.2.2	SD/SDIO Host Controller	278
13.2.3	Data FIFO	279
13.2.4	Command and Control Logic	279
13.2.5	Bus Monitor	279
13.2.6	Stream Write and Read	280
13.2.7	Clocks	280
13.2.8	Soft Resets	280

13.2.9	FIFO Overrun and Underrun Conditions	280
13.3	Protocols	281
13.3.1	Data Transfer Protocol Overview	281
13.3.2	Data Transfers Without DMA	281
13.3.3	Using DMA	284
13.3.4	Using ADMA	287
13.3.5	Abort Transaction	288
13.3.6	External Interface Usage Example	289
13.3.7	Supported Configurations	290
13.3.8	Bus Voltage Translation	291
13.4	SDIO Controller Media Interface Signals	291
13.4.1	SDIO EMIO Considerations	292

Chapter 14: General Purpose I/O (GPIO)

14.1	Introduction	293
14.2	Block Diagram	294
14.3	GPIO Control of Device Pins	295
14.3.1	Special Consideration for EMIO Signals	296
14.3.2	Special Treatment of Bank0[8:7]	297
14.4	Interrupt Function	297

Chapter 15: USB Host, Device, and OTG Controllers

15.1	Introduction	299
15.1.1	Key Features	299
15.1.2	System Viewpoint	300
15.1.3	Notices	300
15.2	Functional Description	301
15.2.1	Block Diagram	301
15.2.2	Control Registers	301
15.2.3	DMA Engine	301
15.2.4	Data Buffers	302
15.2.5	Protocol Engine	302
15.2.6	Transceiver Interface	302
15.2.7	USB Clocks	302
15.3	Technology Description	303
15.3.1	Block Diagram	304
15.3.2	Software Model	304
15.4	Device Data Structure	305
15.5	Host Data Structure	306
15.6	Hardware Model	307
15.6.1	USB-HS Block Diagram	307
15.6.2	DMA Engine	308
15.6.3	Dual Port RAM Controller	308
15.6.4	Protocol Engine	309
15.6.5	Port Controller	310
15.7	Operational Models	310
15.7.1	Host Operational Model	310
15.7.2	EHCI Deviation — Host Mode Only	311

15.7.3	Embedded Transaction Translator Function	311
15.7.4	Embedded Transaction Translator (Multi Port Host Only)	312
15.7.5	Embedded Design Interface	317
15.7.6	Miscellaneous Variations from EHCI	317
15.8	Device Operational Model	318
15.8.1	Device Controller Initialization	319
15.8.2	Port State and Control	320
15.8.3	Operational Model for Packet Transfers	324
15.8.4	Managing Queue Heads	332
15.8.5	Managing Transfers with Transfer Descriptors	334
15.8.6	Servicing Interrupts	337
15.9	OTG Operations	338
15.9.1	Register Bits	338
15.9.2	Hardware Assist	339
15.10	Host Data Structures	341
15.10.1	Periodic Frame List	341
15.10.2	Asynchronous List Queue Head Pointer	343
15.10.3	Isochronous (High-Speed) Transfer Descriptor (iTd)	344
15.10.4	Next Link Pointer	345
15.10.5	iTd Transaction Status and Control List	345
15.10.6	iTd Buffer Page Pointer List (Plus)	347
15.10.7	Split Transaction Isochronous Transfer Descriptor (siTd)	348
15.10.8	Next Link Pointer	349
15.10.9	siTd Endpoint Capabilities/Characteristics	350
15.10.10	siTd Transfer State	351
15.10.11	siTd Buffer Pointer List (plus)	353
15.10.12	siTd Back Link Pointer	354
15.10.13	Queue Element Transfer Descriptor (qTd)	354
15.10.14	Next qTd Pointer	355
15.10.15	Alternate Next qTd Pointer	356
15.10.16	qTd Token	357
15.10.17	qTd Buffer page Pointer List	360
15.10.18	Queue Head	361
15.10.19	Queue Head Horizontal Link Pointer	362
15.10.20	Endpoint Capabilities and Characteristics	362
15.10.21	Transfer Overlay	365
15.10.22	Periodic Frame Span Traversal Node (FSTN)	366
15.10.23	STN Normal Path Pointer	367
15.10.24	FSTN Back Path Link Pointer	368
15.11	Device Data Structures	368
15.11.1	Endpoint Queue Head (qQH)	369
15.11.2	Endpoint Capabilities/Characteristics	370
15.11.3	Transfer Overlay	371
15.11.4	Current qTd Pointer	371
15.11.5	Setup Buffer	371
15.11.6	Endpoint Transfer Descriptor (qTd)	372
15.12	I/O Signals	375

Chapter 16: Gigabit Ethernet Controller

16.1	Introduction	376
16.1.1	Block Diagram	377

16.1.2 Features	377
16.1.3 System Viewpoint	378
16.1.4 Notices	379
16.2 Functional Description and Programming Model.	379
16.2.1 MAC Transmitter	379
16.2.2 MAC Receiver	381
16.2.3 MAC Filtering	382
16.2.4 Wake on LAN Support	385
16.2.5 DMA Block	386
16.2.6 Checksum Offloading	396
16.2.7 IEEE 1588 Time Stamp Unit	398
16.2.8 MAC 802.3 Pause Frame Support	401
16.3 Programming Examples	405
16.3.1 Initialization	405
16.3.2 PHY Maintenance	407
16.3.3 Servicing Interrupts	407
16.3.4 Transmitting Frames	408
16.3.5 Receiving Frames	408
16.4 Register Overview.	409
16.4.1 Control Registers	409
16.4.2 Status and Statistics Registers	410
16.5 Signals and I/O Connections	412
16.5.1 MIO–EMIO Interface Routing	412
16.5.2 Precision Time Protocol	412
16.5.3 Programmable Logic (PL) Implementations	412
16.5.4 Wiring Diagram Example	413
16.5.5 RGMII Interface via MIO	413
16.5.6 GMII/MII Interface via EMIO	414
16.5.7 MDIO Interface Signals via MIO and EMIO	415

Chapter 17: SPI Controller

17.1 Introduction	416
17.1.1 Features	416
17.1.2 System Viewpoint	417
17.2 Functional Description	418
17.2.1 Block Diagram	418
17.2.2 Master Mode	418
17.2.3 Multi-Master Support	419
17.2.4 Slave Mode	420
17.2.5 SPI FIFOs	420
17.2.6 SPI Clocks	421
17.2.7 SPI EMIO Considerations	421
17.3 I/O Interface Signals	421

Chapter 18: CAN Controller

18.1 Introduction	423
18.1.1 Features	423
18.1.2 System Viewpoint	424
18.1.3 Block Diagram	424
18.1.4 Notices	425

18.2 Functional Description	426
18.2.1 Controller Modes	426
18.2.2 Message Format	429
18.2.3 Message Buffering	431
18.2.4 Interrupts	433
18.2.5 Rx Message Filtering	435
18.2.6 Protocol Engine	438
18.3 Programming Guide	440
18.3.1 Overview	440
18.3.2 Configuration Mode State	440
18.3.3 Start-up Controller	441
18.3.4 Change Operating Mode	442
18.3.5 Write Messages to TxFIFO	442
18.3.6 Write Messages to TxHPB	443
18.3.7 Read Messages from RxFIFO	443
18.3.8 Register Overview	444
18.4 System Functions	445
18.4.1 Clocks	445
18.4.2 Resets	446
18.5 I/O Interface	447
18.5.1 MIO Programming	447
18.5.2 MIO-EMIO Signals	448

Chapter 19: UART Controller

19.1 Introduction	449
19.1.1 Features	449
19.1.2 System Viewpoint	450
19.1.3 Block Diagram	451
19.1.4 Notices	451
19.2 Functional Description	452
19.2.1 Baud Rate Generator	452
19.2.2 Transmit FIFO	453
19.2.3 Transmitter Data Stream	454
19.2.4 Receiver FIFO	454
19.2.5 Receiver Data Capture	454
19.2.6 Mode Switch	456
19.2.7 Modem Control	457
19.3 Programming Guide	457
19.3.1 Configuration Register	458
19.3.2 Modem Flow Control	458
19.3.3 Rx and Tx Data Transfers	459
19.3.4 Interrupt Control	461
19.3.5 Register Overview	463
19.4 System Functions	464
19.4.1 Clocks	464
19.4.2 Resets	464
19.5 I/O Interface	465
19.5.1 MIO Programming	465
19.5.2 MIO – EMIO Signals	465

Chapter 20: I2C Controller

20.1 Introduction	467
20.1.1 Features	467
20.1.2 System Block Diagram	468
20.2 Functional Description	469
20.2.1 Block Diagram	469
20.2.2 Master Mode	469
20.2.3 Slave Monitor Mode	471
20.2.4 Slave Mode	471
20.2.5 I2C Speed	473
20.2.6 Multi-Master Operation	473
20.3 Register Overview	474
20.4 Interface Signals	474

Chapter 21: Programmable Logic Description

21.1 Introduction	475
21.1.1 Features	475
21.2 Detailed Description	477
21.2.1 CLBs, Slices, and LUTs	477
21.2.2 Clock Management	477
21.2.3 Block RAM	479
21.2.4 Digital Signal Processing — DSP Slice	480
21.2.5 Input/Output	481
21.2.6 Low-Power Serial Transceivers	482
21.2.7 Integrated Block for PCI Express Designs	483
21.2.8 XADC (Analog-to-Digital Converter)	484
21.2.9 Configuration	485
21.3 PS-PL Interfaces	486

Chapter 22: Programmable Logic Design Guide

22.1 Introduction	487
22.2 Programmable Logic for Software Offload	487
22.2.1 Benefits of Using PL to Implement Software Algorithms	487
22.2.2 Designing PL Accelerators	488
22.2.3 PL Acceleration Limits	489
22.2.4 Power Offload	489
22.2.5 Real Time Offload	490
22.2.6 Reconfigurable Computing	491
22.3 PL and Memory System Performance Overview	492
22.3.1 Theoretical Bandwidth	492
22.3.2 DDR Efficiency	493
22.3.3 OCM Efficiency	494
22.3.4 Interconnect Throughput Bottlenecks	494
22.4 Choosing a Programmable Logic Interface	495
22.4.1 PL Interface Comparison Summary	495
22.4.2 Cortex-A9 CPU via General Purpose Masters	495
22.4.3 PS DMA Controller (DMAC) via General Purpose Masters	496
22.4.4 PL DMA via AXI High-Performance (HP) Interface	497

22.4.5 PL DMA via AXI ACP	498
22.4.6 PL DMA via General Purpose AXI Slave (GP)	499

Chapter 23: Programmable Logic (PL)

Test and Debug

23.1 Introduction	500
23.1.1 Features	500
23.1.2 Block Diagram	501
23.1.3 System Viewpoint	502
23.2 Functional Description	502
23.2.1 Basic Operation	502
23.2.2 Packet Generation	503
23.2.3 Packet Format	505
23.3 Signals	507
23.3.1 General-Purpose Debug Signals	507
23.3.2 Trigger Signals	508
23.3.3 Trace Signals	508
23.4 Register Overview	509
23.5 Programming Model	509
23.5.1 FTM Security	509

Chapter 24: Power Management

24.1 Introduction	510
24.2 Voltage Domains	511
24.3 Features	511
24.4 PS Dynamic Power Reduction in Running Mode	511
24.5 Sleep Mode Sequence	512

Chapter 25: Clocks

25.1 Introduction	513
25.1.1 Block Diagram	513
25.1.2 Clock Generation	514
25.1.3 Reset	514
25.1.4 System Viewpoint	515
25.2 Power Management	516
25.2.1 Top-Level (Central) Interconnect Clock Disable	516
25.3 CPU Clock Domains	517
25.4 Clock Programming Examples	519
25.5 Basic Clock Branch Design	520
25.6 DDR Clock Domains	522
25.7 I/O Peripheral (IOP) Clocks	523
25.7.1 USB Clocks	523
25.7.2 Ethernet Clocks	524
25.7.3 SDIO, SMC, SPI, Quad-SPI and UART Clocks	525
25.7.4 CAN Clocks	525

25.8 GPIO and I2C Clocks	526
25.9 PL Clocks	526
25.10 Miscellaneous Clocks	527
25.10.1 Trace Port Clock	527
25.11 Register Overview	528
25.12 Programming Model	529
25.12.1 Branch Clock Generator	529
25.12.2 DDR Clocks	529
25.12.3 Digitally Controlled Impedance (DCI) Clock	529
25.12.4 PLLs	529
25.12.5 CPU_DIVISOR(ARM_CLK_CTRL[13:8])	531

Chapter 26: Reset System

26.1 Introduction	532
26.1.1 Features	532
26.1.2 Block Diagram	532
26.1.3 Reset Hierarchy	533
26.1.4 Boot Flow	534
26.2 Reset Sources	535
26.2.1 Power-on Reset (PS_POR_B)	535
26.2.2 External System Reset (PS_SRST_B)	536
26.2.3 System Software Reset	536
26.2.4 Watchdog Timer Resets	536
26.2.5 Secure Violation Lock Down	536
26.2.6 Debug Resets	536
26.3 Reset Effects	537
26.3.1 Peripherals	537
26.4 PL Resets	538
26.4.1 PL General Purpose User Resets	538
26.5 Register Overviews	538
26.5.1 Reset Subsystem Control and Status	538
26.5.2 System Reset Control	539
26.5.3 Peripheral Reset Control	539

Chapter 27: JTAG and DAP Subsystem

27.1 Introduction	540
27.1.1 Block Diagram	540
27.1.2 Features	542
27.2 Functional Description	543
27.3 I/O Signals	545
27.4 Programming Model	546
27.4.1 User Case I: Software Debug with Trace Port Enabled	546
27.4.2 User Case II: PS and PL Debug with Trace Port Enabled	546
27.5 ARM DAP Controller	547
27.6 Trace Port Interface Unit (TPIU)	549
27.7 Xilinx TAP Controller	549

Chapter 28: System Test and Debug

28.1 Introduction	551
28.1.1 Features	551
28.1.2 Notices	552
28.2 Functional Description	552
28.2.1 Debug Access Port (DAP)	553
28.2.2 Embedded Cross Trigger (ECT)	554
28.2.3 Program Trace Macrocell (PTM)	555
28.2.4 Instrumentation Trace Macrocell (ITM)	556
28.2.5 Funnel	556
28.2.6 Embedded Trace Buffer (ETB)	557
28.2.7 Trace Packet Output (TPIU)	557
28.3 I/O Signals	558
28.4 Register Overview	559
28.4.1 Memory Map	559
28.4.2 Functionality	560
28.5 Programming Model	563
28.5.1 Authentication Requirements	563

Chapter 29: On-Chip Memory (OCM)

29.1 Introduction	565
29.1.1 Block Diagram	566
29.1.2 Features	566
29.1.3 System Viewpoint	567
29.2 Functional Description	568
29.2.1 Overview	568
29.2.2 Optimal Transfer Alignment	568
29.2.3 Clocking	568
29.2.4 Arbitration Scheme	568
29.2.5 Interrupts	573
29.3 Register Overview	574
29.4 Programming Model	574
29.4.1 Changing Address Mapping	574
29.4.2 AXI Responses	575

Chapter 30: Analog-to-Digital Converter Interface

30.1 Introduction	576
30.1.1 Block Diagram	577
30.1.2 Features	577
30.1.3 Notices	578
30.2 System Viewpoint	578
30.3 Functional Description	578
30.4 Register Overview	580
30.5 Programming Model	581

Chapter 31: PCI Express

31.1 Introduction	582
31.2 Block Diagram	583
31.3 Features.....	583

Chapter 32: Device Secure Boot

32.1 Introduction	585
32.1.1 Block Diagram	585
32.1.2 Features	585
32.2 Functional Description	587
32.2.1 Master Secure Boot	587
32.2.2 External Boot Devices	588
32.2.3 Secure Boot Image	588
32.2.4 eFuse Settings	590
32.2.5 Boot Image and Bitstream Encryption	590
32.2.6 Boot Image and Bitstream Decryption and Authentication	590
32.2.7 HMAC Signature	590
32.2.8 Key Management	591
32.3 Secure Boot Features	591
32.3.1 Non-Secure Boot State	591
32.3.2 Secure Boot State	591
32.3.3 Security Lockdown	591
32.3.4 Golden Image Search	592
32.3.5 JTAG and Debug Considerations	592
32.3.6 Readback	592
32.4 Programming Considerations	592

Appendix A: Additional Resources

A.1 Xilinx Resources	594
A.2 Solution Centers	595
A.3 References	595
A.3.1 Zynq-7000 EPP Documents	595
A.3.2 PL Documents – Device and Boards	595
A.3.3 Advanced eXtensible Interface (AXI) Documents	596
A.3.4 Software Documents	596
A.3.5 git Information	596
A.3.6 Design Tool Documents	596
A.3.7 EDK Documentation	597
A.3.8 Third-Party IP and Standards Documents	597

Appendix B: Register Details

B.1 Overview	599
B.2 Acronyms	600
B.3 Module Summary	601
B.4 AXI FIFO Interface (AFI)	604
B.5 CAN Controller (can).....	615
B.6 DDR Memory Controller (ddrc)	660

B.7 CoreSight Cross Trigger Interface (cti)	734
B.8 Performance Monitor Unit (cortexa9_pmu)	768
B.9 CoreSight Program Trace Macrocell (ptm)	778
B.10 Debug Access Port (dap)	827
B.11 CoreSight Embedded Trace Buffer (etb)	842
B.12 PL Fabric Trace Monitor (ftm)	862
B.13 CoreSight Trace Funnel (funnel)	883
B.14 CoreSight Instrumentation Trace Macrocell (itm)	898
B.15 CoreSight Trace Packet Output (tpiu)	946
B.16 Device Configuration Interface (devcfg)	967
B.17 DMA Controller (dmac)	993
B.18 Gigabit Ethernet Controller (GEM)	1119
B.19 General Purpose I/O (gpio)	1199
B.20 Interconnect QoS (qos301)	1228
B.21 NIC301 Address Region Control (nic301_addr_region_ctrl_registers)	1234
B.22 I2C Controller (IIC)	1236
B.23 L2 Cache (L2Cpl310)	1247
B.24 Application Processing Unit (mpcore)	1286
B.25 On-Chip Memory (ocm)	1341
B.26 Quad-SPI Flash Controller (qspi)	1345
B.27 SD Controller (sdio)	1363
B.28 System Level Control Registers (slcr)	1405
B.29 Static Memory Controller (pl353)	1542
B.30 SPI Controller (SPI)	1570
B.31 System Watchdog Timer (swdt)	1582
B.32 Triple Timer Counter (ttc)	1586
B.33 UART Controller (UART)	1607
B.34 USB Controller (usb)	1623

Introduction

1.1 Overview

The Zynq™-7000 family is based on the Xilinx® Extensible Processing Platform (EPP) architecture. These products integrate a feature-rich dual-core ARM® Cortex™-A9 MPCore™ based processing system (PS) and Xilinx programmable logic (PL) in a single device, built on a state-of-the-art, high-performance, low-power (HPL), 28 nm, and high-k metal gate (HKMG) process technology. The ARM Cortex-A9 MPCore CPUs are the heart of the PS which also includes on-chip memory, external memory interfaces, and a rich set of I/O peripherals.

The Zynq-7000 family offers the flexibility and scalability of an FPGA, while providing performance, power, and ease of use typically associated with ASIC and ASSPs. The range of devices in the Zynq-7000 EPP family enables designers to target cost-sensitive as well as high-performance applications from a single platform using industry-standard tools. While each device in the Zynq-7000 family contains the same PS, the PL and I/O resources vary between the devices. As a result, the Zynq-7000 EPP devices are able to serve a wide range of applications including:

- Automotive driver assistance, driver information, and infotainment
- Broadcast camera
- Industrial motor control, industrial networking, and machine vision
- IP and Smart camera
- LTE radio and baseband
- Medical diagnostics and imaging
- Multifunction printers
- Video and night vision equipment

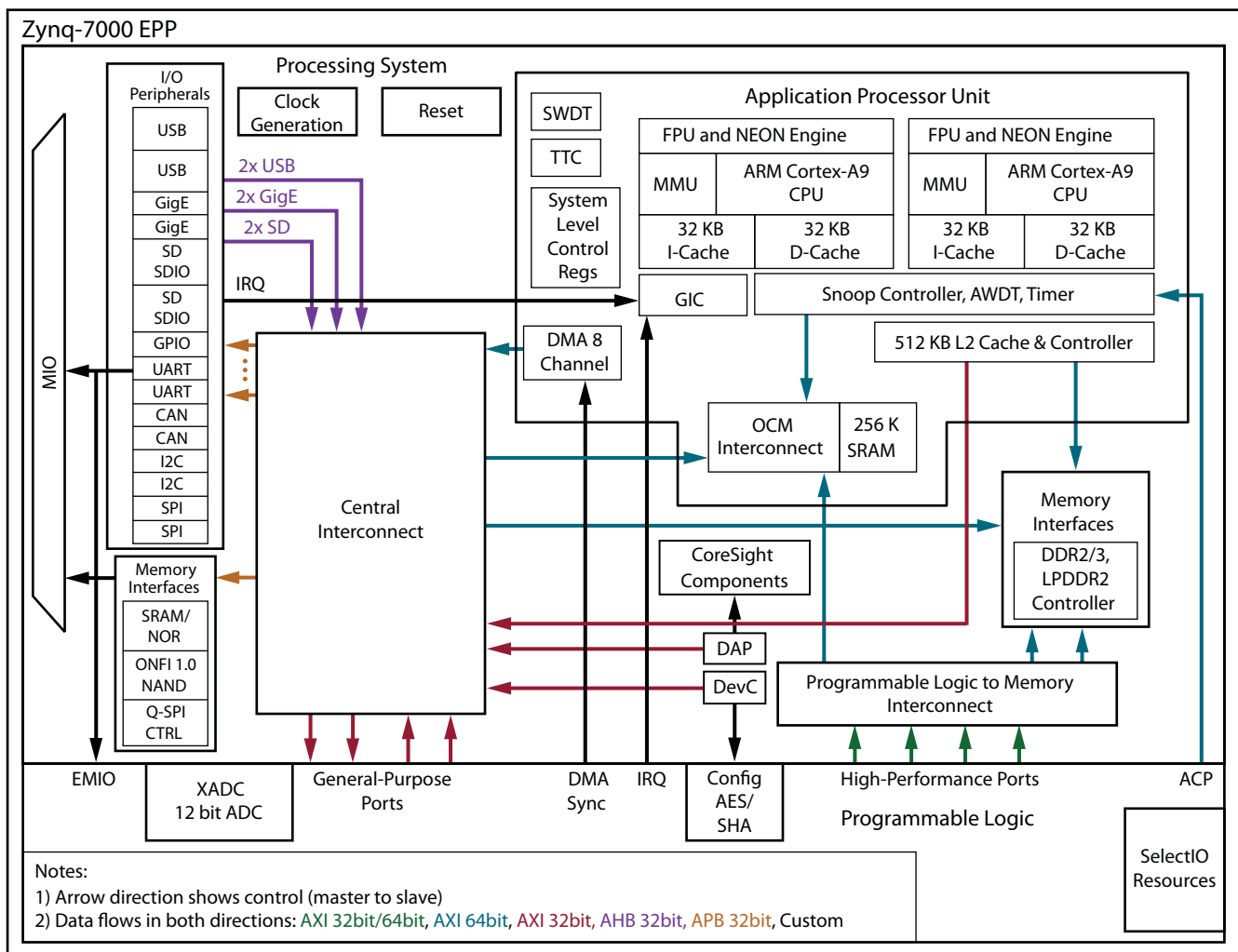
The Zynq-7000 architecture conveniently maps the custom logic and software in the PL and PS respectively. It enables the realization of unique and differentiated system functions. The integration of the PS with the PL provides levels of performance that two-chip solutions (e.g., an ASSP with an FPGA) cannot match due to their limited I/O bandwidth, loose-coupling and power budgets.

Xilinx and the Xilinx Alliance partners offer a large number of soft IP modules for the Zynq-7000 family. Stand-alone and Linux device drivers are available for the peripherals in the PS and the PL from Xilinx and additional OSes and board support packages (BSPs) from partners. The award-winning ISE® Design Suite: Embedded Edition development environment enables a rapid product development for software, hardware, and systems engineers. Many third-party software development tools are also available.

The processors in the PS always boot first, allowing a software centric approach for PL system boot and PL configuration. The PL can be configured as part of the boot process or configured at some point in the future. Additionally, the PL can be completely reconfigured or used with partial, dynamic reconfiguration (PR). PR allows configuration of a portion of the PL. This enables optional design changes such as updating coefficients or time-multiplexing of the PL resources by swapping in new algorithms as needed. This latter capability is analogous to the dynamic loading and unloading of software modules. The PL configuration data is referred to as a bitstream.

1.1.1 Block Diagram

Figure 1-1 illustrates the functional blocks of the Zynq-7000 EPP. The PS and the PL are on separate power domains, enabling the user of these devices to power down the PL for power management if required.



DS190_01_032112

Figure 1-1: Zynq-7000 Extensible Processing Platform Overview

The Zynq-7000 system is composed of the following major functional blocks:

- Processing System (PS)
 - Application processor unit (APU)
 - Memory interfaces
 - I/O peripherals (IOP)
 - Interconnect
- Programmable Logic (PL)

1.1.2 Documentation Resources

The PL is derived from Xilinx's 7 Series FPGA technology (Artix™-7 for the Z-7010/Z-7020 and Kintex™-7 for the Z-7030/Z-7045). The PL is used to extend the functionality to meet specific application requirements. The PL includes many different types of resources including configurable logic blocks (CLBs), port and width configurable block RAM (BRAM), DSP slices with 25 x 18 multiplier, 48-bit accumulator and pre-adder (DSP48E1), a user configurable analog to digital convertor (XADC), clock management tiles (CMT), a configuration block with 256b AES for decryption and SHA for authentication, configurable SelectIO™ and optionally GTX multi-gigabit transceivers and integrated PCI Express® (PCIe) block.

To learn more about the PL resources, refer to the following Xilinx 7 Series FPGA User Guides:

- *UG471, 7 Series FPGAs SelectIO Resources User Guide*
- *UG472, 7 Series FPGAs Clocking Resources User Guide*
- *UG473, 7 Series FPGAs Memory Resources User Guide*
- *UG474, 7 Series FPGAs Configurable Logic Block User Guide*
- *UG476, 7 Series FPGAs GTX Transceiver User Guide*
- *UG477, 7 Series FPGAs Integrated Block v1.3 for PCI Express User Guide*
- *UG479, 7 Series FPGAs DSP48E1 User Guide*
- *UG480, 7 Series FPGAs XADC User Guide*

The PS and PL can be tightly or loosely coupled using multiple interfaces and other signals that have a combined total of over 3,000 connections. This enables the designer to effectively integrate user-created hardware accelerators and other functions in the PL fabric that are accessible to the processors and can also access memory resources in the processing system.

The PS I/O peripherals, including the static/flash memory interfaces share a multiplexed I/O (MIO) of up to 54 MIO pins. Zynq-7000 EPP devices also include the capability to utilize the I/Os that are part of the PL domain for many of the PS I/O Peripherals. This is done through an extended multiplexed I/O interface (EMIO).

The system includes many types of security, test and debug features. The Zynq-7000 EPP can be booted securely or non-securely. The PL configuration bitstream can be applied securely or non-securely. Both of these use the 256b triple-des AES decryption and SHA authentication blocks that are part of the PL. Therefore, to use these security features, the PL must be powered on.

The boot process is multi-stage and minimally includes the Boot ROM and the first-stage boot loader (FSBL). The Zynq-7000 EPP includes a factory-programmed Boot ROM that is not user accessible. The boot ROM determines whether the boot is secure or non-secure, performs some initialization of the system and clean-ups, reads the mode pins to determine the primary boot device and finishes once it is satisfied it can execute the FSBL.

After a system reset, the system automatically sequences to initialize the system and process the first stage boot loader from the selected external boot device. The process enables the user to configure the EPP platform as needed, including the PS and the PL. Optionally, the JTAG interface can be enabled to give the design engineer access to the PS and the PL for test and debug purposes.

Power to the PL can be optionally shut off to reduce power consumption. In addition, the clocks in the PS can be dynamically slowed down or gated off to reduce power further. Zynq-7000 EPP devices support ARM's sleep modes to obtain minimal power drain, but still are able to wake up when certain events occur.

Elements of the Zynq-7000 EPP are described from the point of view of the PS. For example, a general purpose slave interface on the PS to the PL means that the master resides in the PL. A high performance slave interface means the high performance master resides in the PL. A general purpose master interface means the PS is the master and the slave resides in the PL.

1.1.3 Notices

Zynq-7000 EPP Device Family

The PS structure for all Zynq-7000 EPP devices is the same except for the following:

7z010 CLG225 Device

The 7z010 CLG225 device (225 pin package) has a limited number of pins that reduces the capability of the MIO, DDR and XADC subsystems.

- 32 MIO pins, see section [2.4.3 MIO Pin Assignment Considerations](#)
- 16 DDR data, see section [10.1.3 Notices](#)
- Four pairs of XADC signals, see section [30.1.3 Notices](#)

Device Revisions

The visual markings are shown in the data sheet. The 7x010 and 7x020 device markings are shown in the [DS187](#) data sheet. The 7x030 and 7x045 device markings are shown in the [DS191](#) data sheet.

Software can read the following registers in all Zynq-7000 EPP devices to determine silicon revision:

- devcfg.PS_VERSION
- slcr.PSS_IDCODE[IDCODE]

The JTAG interface also includes the IDCODE revision content.

TrustZone Capabilities

TrustZone is hardware that is built into all Zynq-7000 EPP devices. The Technical Reference Manual includes TrustZone hardware descriptions. Please contact your Xilinx Sales Representative about obtaining detailed information related to operating the system in a secure mode.

1.2 Processing System (PS) Features and Descriptions

1.2.1 Application Processor Unit (APU)

The application processor unit (APU) provides an extensive offering of high-performance features and standards-compliant capabilities.

Dual ARM Cortex-A9 MPCore CPUs with ARM v7

- Run-time options allow single processor, asymmetrical (AMP) or symmetrical multiprocessing (SMP) configurations
- ARM Version 7 ISA: Standard ARM instruction set and Thumb®-2, Jazelle® RCT and Jazelle DBX Java™ acceleration
- NEON™ 128b SIMD coprocessor and VFPv3 per MPCore
- 32 KB instruction and 32 KB data L1 caches with parity per MPCore
- 512 KB of shareable L2 cache with parity
- Private timers and watchdog timers

System Features

- System-Level Control Registers (SLCRs)
 - A group of various registers that are used to control the PS behavior
 - The register map is located in [Chapter 4, System Addresses](#)
 - The SLCR registers related to a specific chapter are listed in the register overview table of that chapter and detailed in [Appendix B, Register Details](#)
- Snoop control unit (SCU) to maintain L1 and L2 coherency
- Accelerator coherency port (ACP) from PL (master) to PS (slave)
 - 64b AXI slave port
 - Can access the L2 and the OCM
 - Transactions are data coherent with L1 and L2 caches
- 256 KB of on-chip SRAM (OCM) with parity
 - Dual ported

- Accessible by the CPUs, PL and central interconnect
- At level of L2, but is not cacheable
- DMA controller
 - Four channels for PS (memory copy to/from any memory in system)
 - Four channels for PL (memory to PL, PL to memory)
- General interrupt controller (GIC)
 - Individual interrupt masks and interrupt prioritization
 - Five CPU-private peripheral interrupts (PPI)
 - Sixteen CPU-private software generated interrupts (SGI)
 - Distributes shared peripheral interrupts (SPI) from the rest of the system, PS and PL
 - 20 from the PL
 - Wait for interrupt (WFI) and wait for event (WFE) signals from CPU sent to PL
 - Enhanced security features to support TrustZone™ technology
- Watchdog timer, triple counter/timer

1.2.2 Memory Interfaces

The memory interfaces includes multiple memory technologies.

DDR Controller

- Supports DDR3, DDR2, LPDDR-2
 - Rate is determined by speed and temperature grade of the device
- 16b or 32b wide
 - ECC on 16b
- Uses up to 73 dedicated PS pins
- Modules (no DIMMs)
 - 32b wide: 4 x 8b, 2 x 16b, 1 x 32b
 - 16b wide: 2 x 8b, 1 x 16b
- Autonomous DDR power down entry and exit based on programmable idle periods
- Data read strobe auto-calibration
- Write data byte enables supported for each data beat
- Low latency read mechanism using HPR queue
- Special urgent signaling to each port
- TrustZone regions programmable on 64 MB boundaries
- Exclusive accesses for two different IDs per port (locked transactions are not supported)

DDR Controller Core and Transaction Scheduler

- Transaction scheduling is done to optimize data bandwidth and latency
- Advanced re-ordering engine to maximize memory access efficiency with target of 90% efficiency with continuous read and write and 80% efficiency with random read and write
- Write-read address collision checking that flushes the write buffer
- Obeys AXI ordering rules

Quad-SPI Controller

Key features of the linear Quad-SPI Controller are:

- Single or dual
- 1x and 2x read support
- 100 MHz 32-bit APB 3.0 interface for I/O mode that allows full device operations including program, read and configuration
- 100 MHz 32-bit AXI linear address mapping interface for read operations
- Single chip select line support
- Supports write protection signal
- Four-bit bidirectional I/O signals
- Read speed of x1, x2 and x4
- Write speed of x1 and x4
- Maximum SPI clock at master mode is 100 MHz
- 252-byte entry FIFO depth to improve Quad-SPI read efficiency
- Supports Quad-SPI device up to 128 Mb density
- Supports dual Quad-SPI with two quad-SPI devices in parallel

In addition, the linear address mapping mode features include:

- Supports regular read-only memory access through the AXI interface
- Up to two SPI flash memories
- Up to 16 MB addressing space for one memory and 32 MB for two memories
- AXI read acceptance capability of 4
- Both AXI incrementing and wrapping-address burst read
- Automatically converts normal memory read operation to SPI protocol, and vice versa
- Serial, Dual and Quad-SPI modes

Static Memory Controller (SMC)

Either of the following can be the primary boot device:

- NAND controller

- 8/16-bit I/O width with one chip select signal
- ONFI specification 1.0
- 16-word read and 16-word write data FIFOs
- 8-word command FIFO
- Programmable I/O cycle timing
- ECC assist
- Asynchronous memory operating mode
- Parallel SRAM/NOR controller
 - 8-bit data width with up to 25 address signals
 - Two chip select signals (with 24 address signals)
 - 16-word read and 16-word write data FIFOs
 - 8-word command FIFO
 - Programmable I/O cycle timing on a per chip select basis
 - Asynchronous memory operating mode
 - 8b data width

1.2.3 I/O Peripherals

The I/O Peripherals (IOP) are a collection of industry-standard interfaces for external data communication:

GPIO

- Up to 54 GPIO signals for device pins routed through the MIO
 - Outputs are 3-state capable
- 192 GPIO signals between the PS and PL via the EMIO
 - 64 Inputs, 128 outputs (64 true outputs and 64 output enables)
- The function of each GPIO can be dynamically programmed on an individual or group basis
 - Enable, bit or bank data write, output enable and direction controls
- Programmable Interrupts on individual GPIO basis
 - Status read of raw and masked interrupt
 - Positive edge, negative edge, either edge, high level, low level sensitivities

Gigabit Ethernet Controllers (Two)

- RGMII interface using MIO pins and external PHY
- Additional interface using PL SelectIO and external PHY with additional soft IP in the PL
 - SGMII interface using PL GTX transceivers
- Built-in DMA with scatter-gather

- IEEE 802.3-2008 and IEEE 1588 revision 2.0
- Wake-on capability

USB Controllers: Each as Host, Device or OTG (Two)

- USB 2.0 high speed on-the-go (OTG) dual role USB host controller or USB device controller operation using the same hardware
- MIO pins only (one USB controller is available in the 7x010 device)
- Built-in DMA
- USB 2.0 high speed device
- USB 2.0 high speed host controller
- The USB host controller registers and data structures are EHCI compatible
- Direct support for USB transceiver low pin interface (ULPI). The ULPI module supports 8 bits
- External PHY required
- Support up to 12 endpoints

SD/SDIO Controllers (Two)

- Can be a primary boot device
- Built-in DMA
- Host mode support only
- Support for version 2.0 of SD specification
- Full speed and low speed support
- 1-bit and 4-bit data interface support
- Low speed clock 0–400 KHz
- Support for high speed interface
- Full speed clock 0-50 MHz with maximum throughput at 25 MB/s
- Support for memory, I/O, and combination cards
- Support for power control modes
- Support for interrupts
- 1 KB Data FIFO interface

SPI Controllers (Two): Master or Slave

- Four wire bus: MOSI, MISO, SCLK, SS
- Full-duplex operation offers simultaneous receive and transmit
- Master mode
 - Manual or auto start transmission of data
 - Manual or auto slave select (SS) mode

- Supports up to three slave select lines
- Allows the use of an external peripheral select 3-to-8 decode
- Programmable delays for data transmission
- Slave mode
 - Programmable start detection mode
- Multi-master environment
 - Drives into three-state if not enabled
 - Identifies an error condition if more than one master detected
- Supports 50 MHz maximum external SPI clock rate via MIO
 - 25 MHz maximum via EMIO to PL SelectIO pins
- Selectable master clock reference
- Programmable master baud rate divisor
- Supports 128-byte read and 128-byte write FIFOs
 - Each FIFO is 8-bit wide
- Programmable FIFO thresholds
- Supports programmable clock phase and polarity
- Supports manual or auto start transmission of data
- Software can poll for status or function as interrupt-driven
- Programmable interrupt generation

CAN Controllers (Two)

- Conforms to the ISO 11898 -1, CAN 2.0A, and CAN 2.0B standards
- Supports both standard (11-bit identifier) and extended (29-bit identifier) frames
- Supports bit rates up to 1 Mb/s
- Transmit message FIFO with a depth of 64 messages
- Transmit prioritization through one high-priority transmit buffer
- Support of watermark interrupts for TxFIFO and RxFIFO
- Automatic re-transmission on errors or arbitration loss in normal mode
- Receive message FIFO with a depth of 64 messages
- Acceptance filtering of four acceptance filters
- Sleep mode with automatic wakeup
- Snoop mode
- Loopback mode for diagnostic applications
- Maskable error and status interrupts
- 16-bit time stamping for receive messages
- Readable error counters

UART Controllers (Two)

- Programmable baud rate generator
- 64-byte receive and transmit FIFOs
- 6, 7, or 8 data bits
- 1, 1.5, or 2 stop bits
- Odd, even, space, mark, or no parity
- Parity, framing and overrun error detection
- Line-break generation and detection
- Automatic echo, local loopback, and remote loopback channel modes
- Interrupts generation
- Rx and Tx signals are on the MIO and EMIO interfaces
- Modem control signals: CTS, RTS, DSR, DTR, RI, and DCD are available on the EMIO interface

I2C Controllers (two)

- Supports 16-byte FIFO
- I2C bus specification version 2
- Programmable normal and fast bus data rates
- Master mode
 - Write transfer
 - Read transfer
 - Extended address support
 - Support HOLD for slow processor service
 - Supports TO interrupt flag to avoid stall condition
- Slave monitor mode
- Slave mode
 - Slave transmitter
 - Slave receiver
 - Extended address support
 - Fully programmable slave response address
 - Supports HOLD to prevent overflow condition
 - Supports TO interrupt flag to avoid stall condition
- Software can poll for status or function as interrupt-driven device
- Programmable interrupt generation

PS MIO I/Os

The PS MIO I/O buffers are split into two voltage domains. Within each domain, each MIO is independently programmable.

- Two I/O voltage banks
 - Bank 0 voltage bank consists of pins 0:15
 - Bank 1 voltage bank consists of pins 16:53
- Each MIO pin is individually programmed for voltage signaling
 - 1.8 and 2.5/3.3 volts
 - CMOS single ended or HSTL differential receiver mode

1.3 Programmable Logic Features and Descriptions

The PL provides a rich architecture of user-configurable capabilities.

- Configurable logic blocks (CLB)
 - 6-input look-up tables (LUTs)
 - Memory capability within the LUT
 - Register and shift register functionality
 - Cascadeable adders
- 36 Kb block RAM
 - Dual port
 - Up to 72-bits wide
 - Configurable as dual 18 Kb
 - Programmable FIFO logic
 - Built-in error correction circuitry
- Digital signal processing — DSP48E1 Slice
 - 25 × 18 two's complement multiplier/accumulator high-resolution (48 bit) signal processor
 - Power saving 25-bit pre-adder to optimize symmetrical filter applications
 - Advanced features: optional pipelining, optional ALU, and dedicated buses for cascading
- Clock management
 - High-speed buffers and routing for low-skew clock distribution
 - Frequency synthesis and phase shifting
 - Low-jitter clock generation and jitter filtering
- Configurable I/Os
 - High-performance SelectIO technology

- High-frequency decoupling capacitors within the package for enhanced signal integrity
- Digitally controlled impedance that can be 3-stated for lowest power, high-speed I/O operation
- High range (HR) I/Os support 1.2 V to 3.3 V
- High performance (HP) I/Os support 1.2 V to 1.8 V (Z-7030 and Z-7045 devices)
- Low-power gigabit transceivers (Z-7030 and Z-7045 devices)
 - High-performance transceivers capable of up to 12.5 Gb/s (GTX)
 - Low-power mode optimized for chip-to-chip interfaces
 - Advanced transmit pre- and post-emphasis, and receiver linear (CTLE) and decision feedback equalization (DFE), including adaptive equalization for additional margin
- Analog-to-digital converter (XADC)
 - Dual 12-bit 1 MSPS analog-to-digital converters (ADCs)
 - Up to 17 flexible and user-configurable analog inputs
 - On-chip or external reference option
 - On-chip temperature ($\pm 4^{\circ}\text{C}$ max error) and power supply ($\pm 1\%$ max error) sensors
 - Continuous JTAG access to ADC measurements
- Integrated interface blocks for PCI Express designs (Z-7030 and Z-7045 devices)
 - Compatible to the PCI Express base specification 2.1 with Endpoint and Root Port capability
 - Supports Gen1 (2.5 Gb/s) and Gen2 (5.0 Gb/s) speeds
 - Advanced configuration options, advanced error reporting (AER), and end-to-end CRC (ECRC) advanced error reporting and ECRC features

1.4 Interconnect Features and Description

Zynq-7000 EPP devices use a variety of interconnect technologies, optimized to the specific communication needs of the functional blocks. For more information, refer to the block diagram in [Figure 1-1](#) or a more detailed diagram in [Figure 5-1](#).

1.4.1 PS Interconnect Based on AXI High Performance Data Path Switches

- OCM interconnect
 - Provides access to the 256 KB memory from the central interconnect and the PL
 - CPUs and ACP interfaces have the lowest latency access to OCM via the SCU
- Central interconnect
 - The central interconnect is 64-bits, connecting the IOP and DMA controller to the DDR memory controller, on-chip RAM, and the AXI_GP interfaces (via their switches) for the PL logic

- Connects the local DMA units in the Ethernet, USB and SD/SDIO controllers to the central interconnect
- Connects masters in the PS to the IOP

1.4.2 PS-PL Interfaces

The PS-PL interface contains all the signals available to the PL designer for integrating the PL-based functions and the PS. There are two types of interfaces between the PL and the PS.

1. Functional interfaces which include AXI interconnect, extended MIO interfaces (EMIO) for most of the I/O peripherals, interrupts, DMA flow control, clocks, and debug interfaces. These signals are available for connecting with user-designed IP blocks in the PL.
2. Configuration signals which include the processor configuration access port (PCAP), configuration status, single event upset (SEU) and Program/Done/Init. These signals are connected to fixed logic within the PL configuration block, providing PS control.

AXI functional interfaces:

- AXI_ACP
 - One 64-bit cache coherent master port in the PL
- AXI_HP, four high performance/bandwidth master ports in the PL
 - 32-bit or 64-bit data master interfaces (independently programmed)
 - Efficient resizing in 32-bit slave interface configuration mode
 - Efficient upsizing to 64-bits for aligned 32-bit transfers in 32-bit slave interface configuration mode
 - Automatic expansion to 64-bits for unaligned 32-bit transfers in 32-bit slave interface configuration mode
 - Dynamic command upsizing translation between 32-bit and 64-bit interfaces, controllable via AxCACHE[1]
 - Separate R/W programmable issuing capability for read and write commands
 - Programmable release threshold of write commands
 - Asynchronous clock frequency domain crossing for all AXI interfaces between the PL and PS
 - Smoothing out of “long-latency” transfers using 1 KB (128 by 64 bit) data FIFOs for both reads and writes
 - QoS signaling available from PL ports
 - Command and data FIFO fill-level counts available to the PL
 - Standard AXI 3.0 interfaces supported
 - Large slave interface read acceptance capability in the range of 14 to 70 commands (burst length dependent)
 - Large slave interface write acceptance capability in the range of 8 to 32 commands (burst length dependent)
- AXI_GP, four general purpose ports
 - Two, 32-bit master interfaces

- Two, 32-bit slave interfaces
 - Asynchronous clock frequency domain crossing for all AXI interfaces between the PL and PS
 - Standard AXI 3.0 interfaces supported
 - AXI_ACP
 - 64-bit slave interface
 - Connects to the snoop control unit for cache coherency between the CPUs and the PL
-

1.5 System Software

Xilinx provides device drivers for all of the I/O peripherals. These device drivers are provided in source format and support bare-metal or stand alone and Linux. An example first-stage boot loader (FSBL) is also provided in source code format. The source drivers for stand alone and FSBL are provided as part of the Xilinx IDE Design Suite Embedded Edition. The Linux drivers are provided via the Xilinx Open Source Wiki at <http://wiki.xilinx.com>

Refer to [UG821](#), *Zynq-7000 Software Developers Guide* for additional information.

In addition, Xilinx Alliance Program partners provide system software solutions for IP, middleware, operation systems, etc. Please refer to the Zynq-7000 landing page at <http://www.xilinx.com/zynq> for the latest information.

Signals, Interfaces, and Pins

2.1 Introduction

This chapter identifies the user visible signals and interfaces in Zynq-7000 EPP devices. The interfaces and signals are organized into major groups as shown in [Figure 2-1](#). The Zynq-7000 EPP devices consist of a Processing System (PS) with a Xilinx Artix™-7 or Kintex™-7 based Programmable Logic (PL) block.

2.2 Power Pins

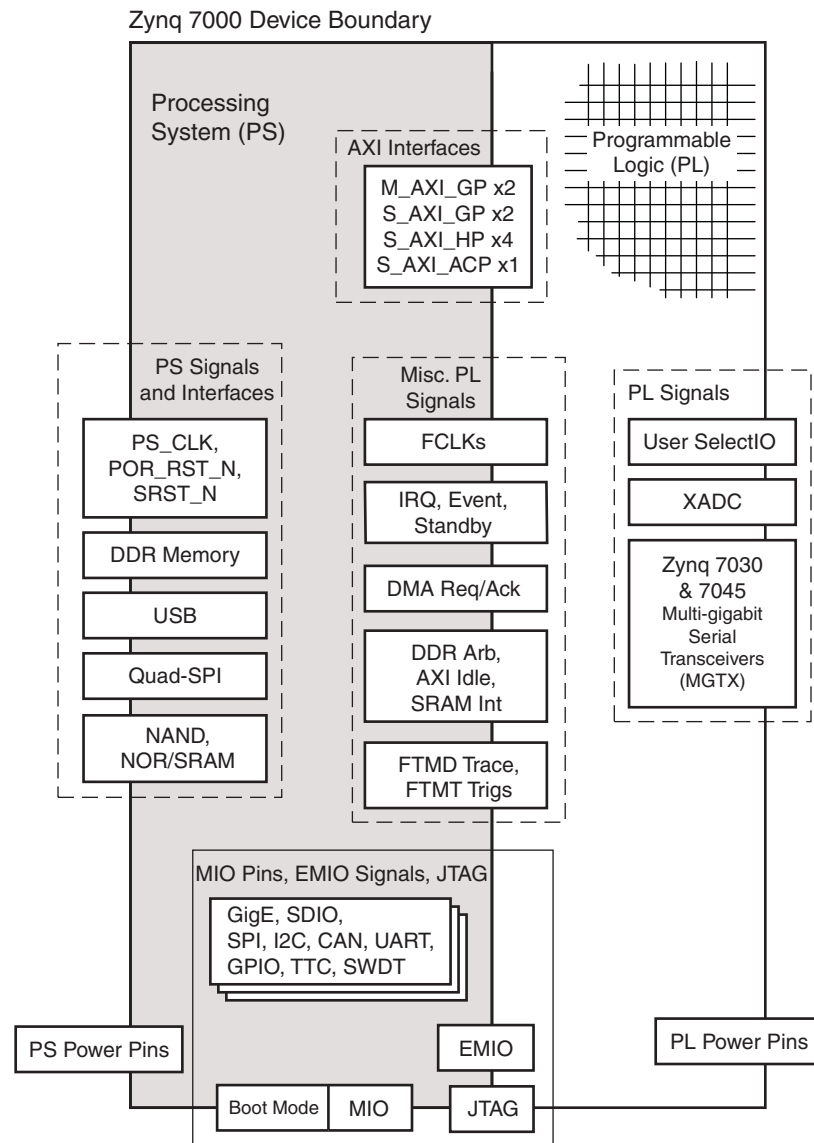
The PS and PL power supplies are fully independent, however the PS power supply must be present whenever the PL power supply is active. The PS includes an independent power supply for the DDR I/O and two independent voltage banks for MIO. The power pins are summarized in [Table 2-1](#). The voltage sequencing and electrical specifications are shown in the applicable Zynq-7000 EPP data sheet. Refer to the applicable Zynq-7000 EPP data sheet and Zynq-7000 EPP packaging and pin documents for more information.

Table 2-1: Power Pins

Type	Pin Name	Nominal Voltage	Power Pin Description
PS Power	V _{CCPINT}	1.0V	Internal logic
	V _{CCPAUX}	1.8V	I/O buffers pre-driver
	V _{CCO_DDR}	1.2V to 1.8V	DDR memory interface
	V _{CCO_MIO0}	1.8V to 3.3V	MIO bank 0, pins 0:15
	V _{CCO_MIO1}	1.8V to 3.3V	MIO bank 1, pins 16:53
	V _{CCPLL}	1.8V	Three PLL clocks, analog
PL Power	V _{CCINT}	1.0V	Internal core logic
	V _{CCAUX}	1.8V	I/O buffer pre-driver
	V _{CCO_#}	1.8V to 3.3V	I/O buffers drivers (per bank)
	V _{CC_BATT}	1.5V	PL decryption key memory backup
	V _{CCBRAM}	1.0V	PL block RAM
	V _{CCAUX_IO_G#}	1.8V to 2.0V	PL auxiliary I/O circuits

Table 2-1: Power Pins (Cont'd)

Type	Pin Name	Nominal Voltage	Power Pin Description
XADC	VCCADC, GNDADC	N/A	Analog power and ground.
Ground	GND	Ground	Digital and analog grounds



UG585_c2_01_042312

Figure 2-1: Signals, Interfaces, and Pins

2.3 PS I/O Pins

A summary of the dedicated PS signal pins is shown in [Table 2-2](#). Refer to the applicable Zynq-7000 EPP data sheet and Zynq-7000 EPP packaging and pin documents for more information.

Table 2-2: PS Signal Pins

Group	Name	Type	Voltage Node	Description
Clock	PS_CLK	I	V _{CCO_MIO0}	System reference clock. See Chapter 25, Clocks .
Reset	PS_POR_B	I	V _{CCO_MIO0}	Power on reset, active low. See Chapter 26, Reset System .
	PS_SRST_B	I	V _{CCO_MIO1}	Debug system reset, active Low. Forces the system to enter a reset sequence. See Chapter 26, Reset System .
MIO	PS_MIO[15:0]	I/O	V _{CCO_MIO0}	See section 2.4 MIO-EMIO .
	PS_MIO[53:16]	I/O	V _{CCO_MIO1}	See section 2.4 MIO-EMIO .
	PS_MIO_VREF	Ref	V _{CCO_MIO1}	Voltage reference for RGMII differential input receivers, normally tied to a voltage node at ½ V _{CCO_MIO1} voltage pin.
DDR	PS_DDR_xxx	I/O	V _{CCO_DDR}	See Chapter 10, DDR Memory Controller .
	PS_DDR_VR[N,P]	N/A	~	DDR DCI voltage reference pins: Externally, VRN is pulled High and VRP is pulled Low with resistors that are normally twice the trace impedance (2 x 40Ω = 80Ω).
	PS_DDR_VREF	Ref	~	Voltage reference for DDR DQ and DQS differential input receivers.

2.4 MIO-EMIO

The MIO is fundamental to the I/O peripheral connections due to the limited number of MIO pins. Software programs the routing of the I/O signals to the MIO pins. The I/O peripheral signals can also be routed to the PL (including PL device pins) through the EMIO interface. This is useful to gain access to more device pins (PL pins) and to allow an I/O peripheral controller to interface to user logic in the PL. See [Figure 2-2](#).

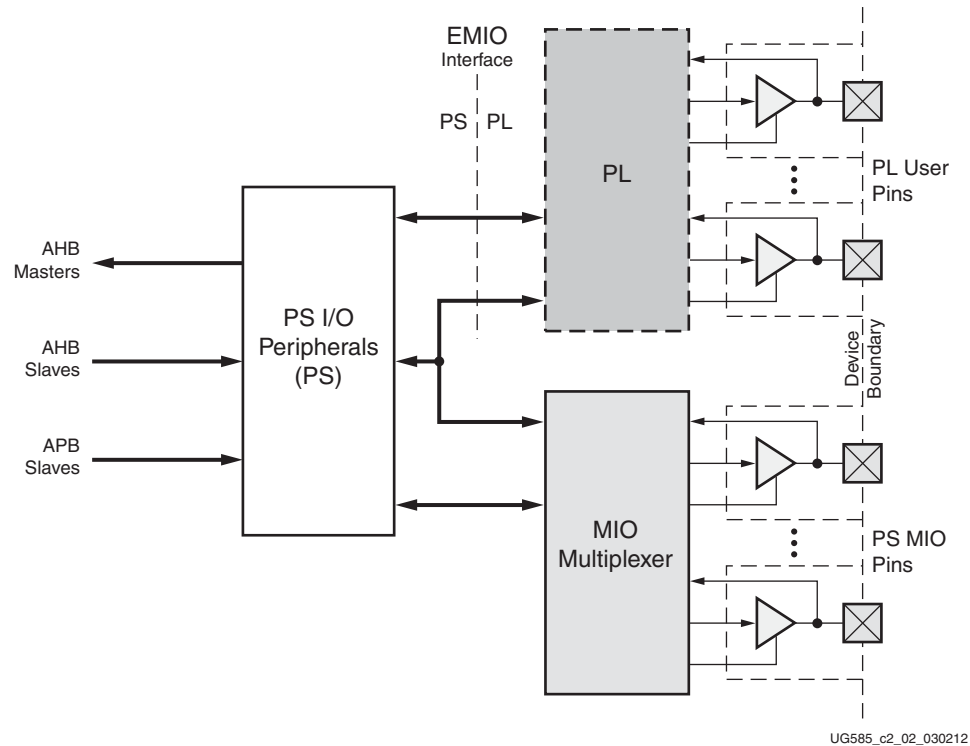


Figure 2-2: MIO-EMIO Overview

2.4.1 I/O Peripheral (IOP) Interface Routing

The I/O multiplexing of the I/O controller signals differs; that is, some IOP signals are solely available on the MIO pin interface. Most of the signals are available via MIO or EMIO. Some of the interface signals are only accessible via EMIO. The routing capabilities for each I/O peripheral is shown in [Table 2-3](#), The MIO pin assignment possibilities are illustrated in [section 2.4.4 MIO-at-a-Glance Table](#).

Table 2-3: I/O Peripheral MIO-EMIO Interface Routing

Peripheral	MIO Routing	EMIO Routing	Cross Reference
TTC [0,1]	Clock In, Wave Out. One pair of signals from each counter.	Clock In, Wave Out. Three pairs of signals from each counter.	See Chapter 8, Timers
SWDT	Clock In, Reset Out	Clock In, Reset Out	See Chapter 8, Timers
SMC	Parallel NOR/SRAM and NAND Flash	Not available	See Chapter 11, Static Memory Controller
Quad-SPI [0,1]	Serial, dual and quad modes	Not available	See Chapter 12, Quad-SPI Flash Controller
SDIO [0,1]	50 MHz	25 MHz	See Chapter 13, SD/SDIO Peripheral Controller

Table 2-3: I/O Peripheral MIO-EMIO Interface Routing (Cont'd)

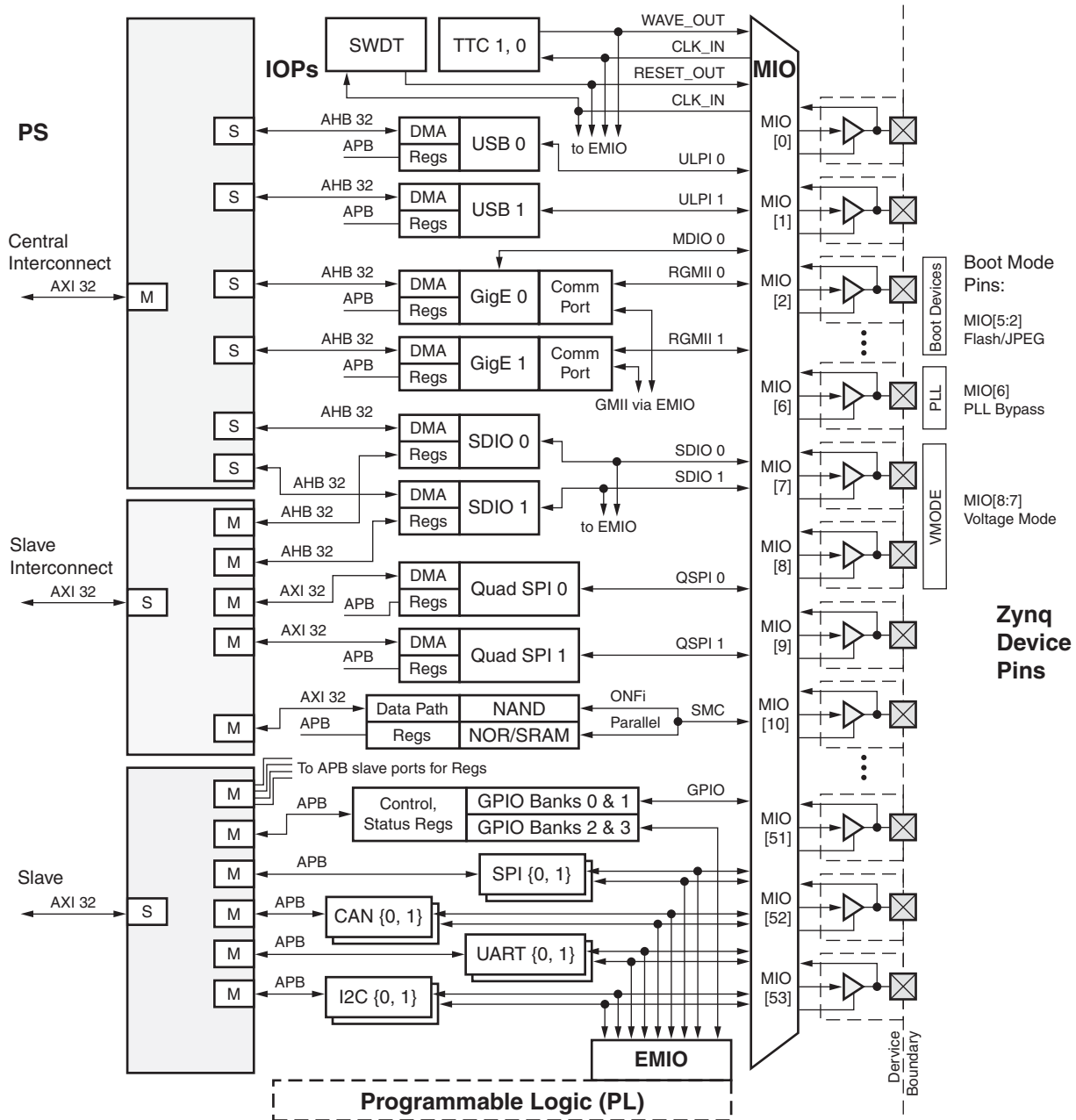
Peripheral	MIO Routing	EMIO Routing	Cross Reference
GPIOs	Up to 54 I/O channels (GPIO Banks 0 and 1)	64 GPIO channels with input, output, 3-state control (GPIO banks 2 and 3)	See Chapter 14, General Purpose I/O (GPIO)
USB [0,1]	Host, device, and OTG	Not available	See Chapter 15, USB Host, Device, and OTG Controllers
Ethernet [0,1]	RGMII v2.0	GMII, RGMII v2.0, RGMII v1.3, RMII, MII, SGMII.	See Chapter 16, Gigabit Ethernet Controller
SPI [0,1]	50 MHz	Available	See Chapter 17, SPI Controller
CAN [0,1]	ISO 11898 -1, CAN 2.0A/B	Available	See Chapter 18, CAN Controller
UART [0,1]	Simple UART: Two pins (TX/RX)	TX, RX, DTR, DCD, DSR, RI, RTS and CTS	See Chapter 19, UART Controller
I2C [0,1]	SCL, SDA {0, 1}	SCL, SDA {0, 1}	See Chapter 20, I2C Controller
PJTAG	TCK, TMS, TDI, TDO	TCK, TMS, TDI, TDO, 3-state for TDO	See Chapter 27, JTAG and DAP Subsystem
Trace Port IU	Up to 16-bit data	Up to 32-bit data	See Chapter 28, System Test and Debug

2.4.2 IOP Interface Connections

For most peripherals, there is flexibility in where the I/O signals can be mapped. The routing capabilities are shown in [Figure 2-4](#). For example, the XPS design software includes up to 12 possible MIO port mappings for CAN, or, if selected, a path to the EMIO interface. The peripheral system connection diagram is shown in [Figure 2-3](#).

The majority of the I/O signals for PS peripherals, other than USB, can be routed to either the PS pins through the MIO, or to the PL pins through the EMIO. Most peripherals also maintain the same protocol between MIO and EMIO, except Gigabit Ethernet. To reduce pin count, a 4-bit RGMII interface runs through the MIO at a 250 MHz data rate. The route through the EMIO includes an 8-bit GMII interface running at a 125 MHz data rate. The USB, Quad-SPI, and SMC interfaces are not available to the EMIO interface to the PL. The PL level shifters must be enabled via LVL_SHFTR_EN before EMIO communication can occur.

On the interconnect side, the USB, Ethernet and SDIO peripherals are connected to the central interconnect to service the six DMA masters. Software accesses the slave-only Quad-SPI and SMC peripherals via the AHB interconnect. The GPIO, SPI, CAN, UART, and I2C slave-only controllers are accessed via the APB bus. All control and status registers are also accessed via the APB interconnect except for the SDIO controllers which each have two AHB interfaces. This architecture is designed to balance the bandwidth needs of each controller interface.



UG585_c2_03_071012

Figure 2-3: I/O Peripherals System Diagram

2.4.3 MIO Pin Assignment Considerations

Normally, each pin is assigned to one function. There are some exceptions as discussed below. When using the EMIO as an alternative to route a signal, be aware that the maximum clocking frequency might be reduced. (The IOP routing is shown in Table 2-3.)

Interface Frequencies: The clocking frequency for each interface routed via MIO is specified in the applicable Zynq-7000 EPP data sheet.

Two MIO Voltage Banks: The MIO pins are split across two independently configured sets of I/O buffers: Bank 0, MIO[15:0] and Bank 1, MIO[53:16]. The signalling voltage is initially configured using the VMODE boot mode strapping pins. Each bank can be configured for 1.8V signalling or 2.5V/3.3V.

Boot Mode Strapping Pins: These pins can be assigned to I/O peripherals in addition to functioning as boot mode pins. MIO pins [8:2], define the boot device, the initial PLL clock bypass mode, and the voltage mode (VMODE) for the MIO banks. The strapping pins are sampled a few PS_CLK clock cycles after the PS_POR_B reset signal de-asserts. The board design ties these signals to VCC or ground using 20 K Ω pull-up and pull-down resistors. More information about the boot mode pin settings is provided in [Chapter 6, Boot and Configuration](#).

I/O Buffer Output Enable Control: The output enable for each MIO I/O buffer is controlled by a combination of the setting of the slcr.MIO_PIN[TRI_ENABLE] register bit, the slcr.MIO_MST_TRI register bits, the selected signal type (input-only or not) and the state of the peripheral controller. The I/O buffer output is enabled when the following condition is met:

- (slcr.MIO_PIN_xx[TRI_ENABLE] = 0) and
- (slcr.MIO_MST_TRIx[PIN_xx_TRI] = 0) and
- (Signal is an output-only or the I/O peripheral desires to drive a signal configured as I/O)

Boot from SDIO: The BootROM expects the SDIO device to be connected to MIO pins 40 through 45 (SDIO 0).

Static Memory Controller (SMC) Interface: Only one SMC interface can be used in a design. The SMC controller consumes many of the MIO pins and neither of the SMC interfaces can be routed to the EMIO.

For example, if an 8-bit NAND Flash is implemented, then Quad-SPI, is not available and the test port is limited to 8-bits. If a 16-bit NAND Flash is implemented, then additional pins are consumed. Ethernet 0 is not available. The SRAM/NOR interface consumes up to 70% of the MIO pins, eliminating Ethernet and USB 0.

The following SRAM/NOR pins are optional: Busy and the upper address bits, as appropriate for the attached device. Also note that the SMC interface straddles the two MIO voltage banks.

Quad-SPI Interface: The lower memory Quad-SPI interface (QSPI_0) must be used if the Quad-SPI memory subsystem is to be used. The upper interface (QSPI_1) is optional and is only used for a two-memory arrangement (parallel or stacked). Do not use the Quad-SPI 1 interface alone.

MIO Pins [8:7] are Outputs: These MIO pins are available as output only. GPIO channels 7 and 8 can only be configured as outputs.

MIO Pins in 7z010 CLG225 Device: This device has 32 MIO pins, 0:15, 28:39, 48, 49, 52, and 53. All other Zynq-7000 EPP devices include all 54 MIO pins and all devices have the same EMIO interface functionality. Refer to section [1.1.3 Notices](#).

The 32 MIO pins available in the 7z010 CLG225 device restrict the functionality of the PS:

- Either one USB or one Ethernet controller via MIO
- No boot from SDIO
- No NOR/SRAM interfacing

- Width of NAND Flash limited to 8 bits

The faded boxes in [Table 2-4](#) represent signals that are not usable in the 7x010 CLG225 device.

2.4.4 MIO-at-a-Glance Table

Table 2-4 presents MIO information in a compact format for easy reference.

Table 2-4: MIO-at-a-Glance

[illegible]

2.4.5 MIO Signal Routing

Signal routing through the MIO is controlled by the MIO_PIN_[53:0] configuration registers located in the slcr registers set. The MIO multiplexes and de-multiplexes the various input and output signals to the MIO pins using four stages of multiplexing, as shown in Figure 2-4. The high-speed data signals (such as RGMII for Gigabit Ethernet and ULPI for USB) are routed through only one multiplexer stage. The slower signals (such as the UART and I2C ports) are routed through all four multiplexer stages. The routing for each MIO pin is independently controlled by multiple bit fields in each MIO_PIN register.

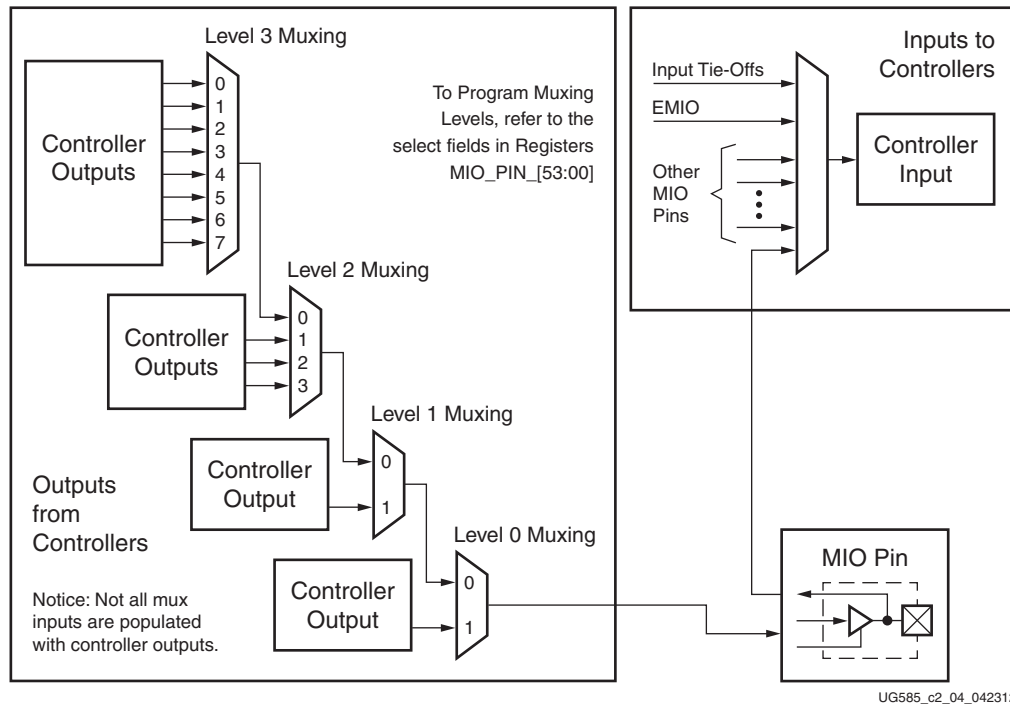


Figure 2-4: MIO Signal Routing

Any of the MIO pins can be programmed to be an external CAN controller reference clock using the CAN_MIOCLK_CTRL register.

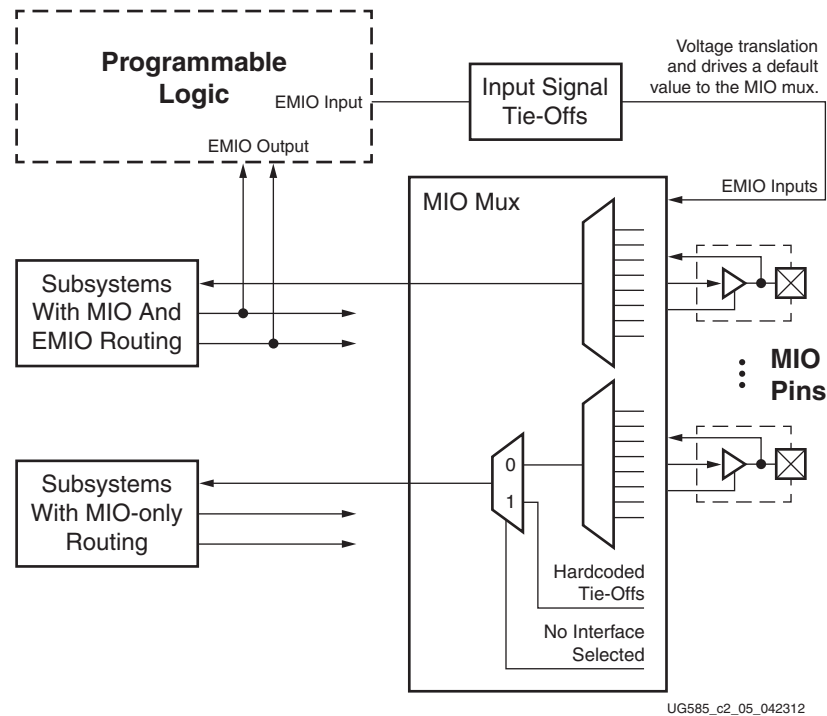
2.4.6 Default Logic Levels

The inputs to the I/O peripherals are driven with default values when another source is not routed to either the MIO or the EMIO. If an input is routed to EMIO, but the PL is powered down, then the same default value is driven to the I/O peripheral. (See Figure 2-5.)

For MIO-only signals, the default signal input is driven when the MIO multiplexer does not route the signal to an MIO pin.

For MIO-EMIO signals, the default signal input is driven when the MIO multiplexer does not route the signal to an MIO pin (the signal defaults to the EMIO interface) and when the signal is programmed to be routed through the EMIO, but the PL either does not drive the signal (not configured) or is not able to drive it (powered down).

The default input signal logic levels are designed to be benign to the I/O peripheral. As a precaution, the related peripheral core should also be disabled when not in use. The logic levels are shown in the signal tables in each chapter for each I/O peripheral.



UG585_c2_05_042312

Figure 2-5: Non-selected Controller Inputs

2.4.7 MIO Pin Electrical Parameters

The MIO_PIN registers include bit fields to control the electrical pin characteristics of each pin. This includes I/O buffer signaling voltage, drive strength, 3-state control, pull-up resistor, and HSTL enable. These are summarized in Table 2-5. Refer to the applicable Zynq-7000 EPP data sheet for electrical specifications.

Table 2-5: MIO I/O Buffer Programmable Parameters

I/O Buffer Parameter	MIO_PIN Register Bit Field	Selections	Comments
Signaling	I/O_Type	LVTTL, LVCMOS (18, 25, 33), HSTL	Selects the drive and receiver type
HSTL Receiver	DisableRcvr	Enable, Disable	Enable when IO_Type = HSTL
Drive Strength	Speed	Fast, Slow	Selects edge rate for all I/O types
3-State Control	3-State Control	Enable, Disable	Enables 3-state for all I/O types
Pull-up	Pull-up	Enable, Disable	Enables pull-up for all I/O types

2.4.8 PS–PL Voltage Level Shifter Enables

The level shifter enables are split into input and output signal groups. If the PL is not powered, the level shifters must be set to 0x0. If the PL is being configured, the level shifters should be set to 0xA to disable the input signals and prevent glitches from propagating into the PS. Once the PL is fully configured, the level shifter enables should be set to 0xF.

2.5 PL AXI Interfaces

The PL AXI interfaces are based on the AXI 3 interface specification. Each interface consists of multiple AXI channels. The interfaces are summarized in [Table 2-6](#). Over a thousand signals are used to implement these nine PL AXI interfaces.

Note: The PL level shifters must be enabled via LVL_SHFTR_EN before PL logic communication can occur.

Table 2-6: PL AXI Interfaces

Interface Name	Interface Description	Master	Slave	Signals
M_AXI_GP0	General Purpose (AXI_GP)	PS	PL	See Chapter 5, Interconnect
M_AXI_GP1		PS	PL	
S_AXI_GP0	General Purpose (AXI_GP)	PL	PS	See Chapter 5, Interconnect
S_AXI_GP1		PL	PS	
S_AXI_ACP0	Accelerator Coherent Port, cache-coherent transaction (ACP)	PL	PS	See Chapter 3, Application Processing Unit
S_AXI_HP0	High Performance ports (AXI_HP) with read/write FIFOs and two dedicated memory ports on DDR controller and a path to the OCM. The AXI_HP interfaces are known also as AFI.	PL	PS	See Chapter 5, Interconnect
S_AXI_HP1		PL	PS	
S_AXI_HP2		PL	PS	
S_AXI_HP3		PL	PS	

2.6 PL–PS Signals

The programmable logic interface group contains miscellaneous interfaces between PS and the PL. An input is driven by the PL and an output is driven by the PS. Signals might have suffixes where an 'N' suffix indicates an active Low signal; otherwise the signal is active High. A 'TN' suffix indicates an active Low 3-state enable signal and is an output to the PL. Output signals to the PL are always driven to either a High or Low level state.

PL-PS signal groups are identified in [Table 2-7](#).

Table 2-7: PL-PS Signal Groups

PL-PS Signal Group	Signal Name	Reference
PL Clocks and Resets	FCLKx	2.6.1 Clocks and Resets
Interrupts	IRQF2Px and IRQP2Fx	2.6.2 Interrupt Signals
Events	EVENTx	2.6.3 Event Signals
IdleAXI, DDR ARB, SRAM Interrupt, FPGA	FPGA, DDR, EMIO	2.6.4 Idle AXI, DDR Urgent/Arb, SRAM Interrupt Signals
DMA Controller	DMACx	2.6.5 DMA Req/Ack Signals

Note: The PL level shifters must be enabled via LVL_SHFTR_EN before PL logic communication can occur.

2.6.1 Clocks and Resets

The PS clock module provides four frequency-programmable clocks (FCLKs) to the PL that are physically spread out along the PS–PL boundary. The clocks can also be individually controlled. The FCLK clocks can be routed to PL clock buffers to serve as a frequency source.

Note: There is no guaranteed timing relationship between the four PL clocks. The FCLK clocks are described in [Chapter 25, Clocks](#).

The PS reset module provides four programmable reset signals to the PL. These signals are asynchronous to the FCLK clocks and are controlled by writing to slcr.FPGA_RST_CTRL SLCR [FPGA[3:0]_OUT_RST]. The resets are described in [Chapter 26, Reset System](#).

These clocks and resets are summarized in [Table 2-8](#). The FCLKCLKTRIGN[3:0] signals are currently not supported. They must be tied to 4'b0000.

Table 2-8: PL Clock and Reset Signals

Type	PL Signal Name	I/O	Reference
PL Clocks	FCLKCLK[3:0]	O	Chapter 25, Clocks
PL Clock Throttle Control	FCLKCLKTRIG [3:0]	I	
PL Resets	FCLKRESETN [3:0]	O	Chapter 26, Reset System

2.6.2 Interrupt Signals

The interrupts from the processing system I/O peripherals (IOP) are routed to the PL and assert asynchronously to the FCLK clocks. In the other direction, the PL can asynchronously assert up to 20 interrupts to the PS. Sixteen of these interrupt signals are mapped to the interrupt controller as a peripheral interrupt where each interrupt signal is set to a priority level and mapped to one or both of the CPUs. The remaining four PL interrupt signals are inverted and routed to the nFIQ and nIRQ interrupt directly to the signals to the private peripheral interrupt (PPI) unit of the interrupt controller. There is an nFIQ and nIRQ interrupt for each of two CPUs. The PS to PL and PL to PS interrupts are listed in [Table 2-9](#). Details of the interrupt signals are described in [Chapter 7, Interrupts](#).

Table 2-9: PL Interrupt Signals

Type	PL Signal Name	I/O	Destination
PL to PS Interrupts	IRQF2P[7:0]	I	SPI: Numbers [68:61].
	IRQF2P[15:8]	I	SPI: Numbers [91:84].
	IRQF2P[19:16]	I	PPI: nFIQ, nIRQ (both CPUs).
PS to PL Interrupts	IRQP2F[27:0]	O	PI Logic. These signals are received from the I/O peripherals and are forwarded to the interrupt controller. These signals are also provided as outputs to the PL.

2.6.3 Event Signals

The PS supports processor events to and from the PL (see [Table 2-10](#)). These signals are asynchronous to the PS and FCLK clocks. For details on these signals, see [Chapter 3, Application Processing Unit](#).

Table 2-10: PL Event Signals

Type	PL Signal Name	I/O	Description
Events	EVENTEVENTI	I	Causes one or both CPUs to wakeup from a WFE state.
	EVENTEVENTO	O	Asserted when one of the CPUs has executed the SEV instruction.
Standby	EVENTSTANDBYWFE[1:0]	O	CPU standby mode: asserted when a CPU is waiting for an event.
	EVENTSTANDBYWFI[1:0]	O	CPU standby mode: asserted when a CPU is waiting for an interrupt.

2.6.4 Idle AXI, DDR Urgent/Arb, SRAM Interrupt Signals

The idle AXI signal to the PS is used to indicate that there are no outstanding AXI transactions in the PL. Driven by the PL, this signal is one of the conditions used to initiate a PS bus clock shut-down by ensuring that all PL bus devices are idle.

The DDR urgent/arb signal is used to signal a critical memory starvation situation to the DDR arbitration for the four AXI ports of the PS DDR memory controller.

Table 2-11: PL AXI Idle, DDR Urgent/Arb and SRAM Interrupt Signals

Type	PL Signal Name	I/O	Destination	Reference
Idle PL AXI interfaces	FPGAIDLEN	I	Central interconnect clock disable logic	
DDR Urgent Signal	DDRARB[3:0]	I	DDR memory controller	Chapter 10, DDR Memory Controller
SRAM	EMIOSRAMINTIN	I		

2.6.5 DMA Req/Ack Signals

There are four sets of DMA controller flow control signals for use by up to four PL slaves connected via the M_AXI_GP interfaces (see [Table 2-11](#)). These four sets of flow control signals correspond to DMA channels 4 through 7, see [Chapter 9, DMA Controller](#).

Table 2-12: PL DMA Signals

Type	Signal	PL Signal Name	I/O	Reference
Clock and Reset	Clock	DMA[3:0]ACLK	I	Chapter 25, Clocks
	Reset	DMA[3:0]RSTN	O	Chapter 26, Reset System
Request	Ready	DMA[3:0]DRREADY	O	Chapter 9, DMA Controller
	Valid	DMA[3:0]DRVALID	I	
	Type	DMA[3:0]DRTYPE[1:0]	I	
	Last	DMA[3:0]DRLAST	I	
Acknowledge	Ready	DMA[3:0]DAREADY	I	
	Valid	DMA[3:0]DAVALID	O	
	Type	DMA[3:0]DATYPE[1:0]	O	

2.7 PL I/O Pins

A summary of the PL I/O pins is shown in [Table 2-13](#). Refer to the applicable Zynq-7000 EPP data sheet and Zynq-7000 EPP packaging and pin documents for more information.

For more information on multi-gigabit serial transceivers pins, see the Pin Description and Design Guidelines section in [UG476, 7 Series FPGAs GTX Transceivers User Guide](#). (Four to sixteen transceivers are available in the Kintex-based Zynq XC7Z030 and XC7Z045 devices.)

Table 2-13: PL Pin Summary

Group	Name	Type	Description
User I/O Pins	IO_LXXY_#, IO_XX_#	I/O	Most user I/O pins are capable of differential signaling and can be implemented as pairs. The top and bottom I/O pins are always single ended.

Table 2-13: PL Pin Summary (Cont'd)

Group	Name	Type	Description
Multi-Gigabit Serial Transceivers	MGTXR[X[P,N]	I	Differential receive and transmit ports. Multi-Gigabit Serial Transceiver pins. Four and 16 transceivers are available in Zynq XC7Z030 and XC7Z045 devices, respectively.
	MGTXTX[P,N]	O	
	MGTAVCC_G#	I	1.0V analog power-supply pin for receiver and transmitter internal circuits.
	MGTAVTT_G#	I	1.2V analog power-supply pin for the transmit driver.
	MGTVCCAUX_G#	I	1.8V auxiliary analog Quad PLL voltage supply for the transceivers.
	MGTREFCLK0/1P	I	Positive differential reference clock for the transceivers.
	MGTREFCLK0/1N	I	Negative differential reference clock for the transceivers.
	MGTAVTTRCAL	N/A	Precision reference resistor pin for internal calibration termination.
	MGTRREF	I	Precision reference resistor pin for internal calibration termination.
PL JTAG	PL_TCK, PL_TMS, PL_TDI, PL_TDO	I/O	See Chapter 27, JTAG and DAP Subsystem .
Configuration	DONE, INIT_B, PROGRAM_B	I/O	See Chapter 21, Programmable Logic Description .
	CFGBVS	I	Pre-configuration I/O standard type for the dedicated configuration bank 0.
	PUDC_B	I	Active Low input enables internal pull-ups during configuration on all SelectIO pins.
XADC	VP, VN	I	Dedicated differential analog inputs.
	VREFP, VREFN	N/A	Reference input (1.25V) and ground.
	AD[15:0]P, AD[15:0]N	I	16 differential auxiliary analog inputs.
Multi-function	MRCC	I	Clock capable I/Os driving BUFRs, BUFIOs, BUFGs and MMCMs/PLLs. In addition, these pins can drive the BUFMR for multi-region BUFIO and BUFR support. These pins become regular user I/Os when not needed as a clock.
	SRCC	I	Clock capable I/Os driving BUFRs, BUFIOs and MMCMs/PLLs. These pins become regular user I/Os when not needed for clocks.
	T[3:0]	I	Four memory byte groups.
	T[3:0]_DQS	I	DDR DQS strobe pin that belongs to the memory byte group T0-T3.
Temperature	DXP, DXN	I	Temperature-sensing diode pins.
Reserved	RSVDVCC	I	Tie to V _{CCO} .
	RSVDGND	I	Tie to ground.

Application Processing Unit

3.1 Introduction

3.1.1 Basic Functionality

The application processing unit (APU), located within the PS, contains two ARM Cortex A9 processors with NEON co-processors that are connected in an MP configuration sharing a 512 KB L2 cache. Each processor is a high-performance and low-power core that implements two separate 32 KB L1 caches for instruction and data. The Cortex-A9 processor implements the ARMv7-A architecture with full virtual memory support and can execute 32-bit ARM instructions, 16-bit and 32-bit Thumb instructions, and 8-bit Java™ byte codes in the Jazelle state. The NEON coprocessor's media and signal processing architecture adds instructions that target audio, video, image and speech processing, and 3D graphics. These advanced single instruction multiple data (SIMD) instructions are available in both ARM and Thumb states. A block diagram of the APU is shown in [Figure 3-1](#).

The two Cortex A9 processors within the APU are organized in an MP configuration with a snoop control unit (SCU) responsible for maintaining L1 cache coherency between the two processors and the ACP interface from the PL. To increase performance, there is a shared unified 512 KB level-two (L2) cache for instruction and data. In parallel to the L2 cache, there is a 256 KB on-chip memory (OCM) module that provides a low-latency memory.

An accelerator coherency port (ACP) facilitates communication between the programmable logic (PL) and the APU. This 64-bit AXI interface allows the PL to implement an AXI master that can access the L2 and OCM while maintaining memory coherency with the CPU L1 caches.

The unified 512 KB L2 cache is 8-way set-associative and allows users to lock the cache content on a line, way, or master basis. All accesses through the L2 cache controller can be routed to the DDR controller or can be sent to other slaves in the PS or PL depending on their address. To reduce latency to the DDR memory, there is a dedicated port from the L2 controller to the DDR controller.

Debug and trace capability is built into the two processor cores and interconnects as a part of the CoreSight debug and trace system. The user can control and interrogate both processors and the memory through the debug access port (DAP). Furthermore, 32-bit AMBA trace bus (ATB) masters from the two processors are funneled with other ATB masters, such as ITM and FTM, to generate the unified PS trace through the on-chip embedded trace buffer (ETB) or the trace-port interface units (TPIU).

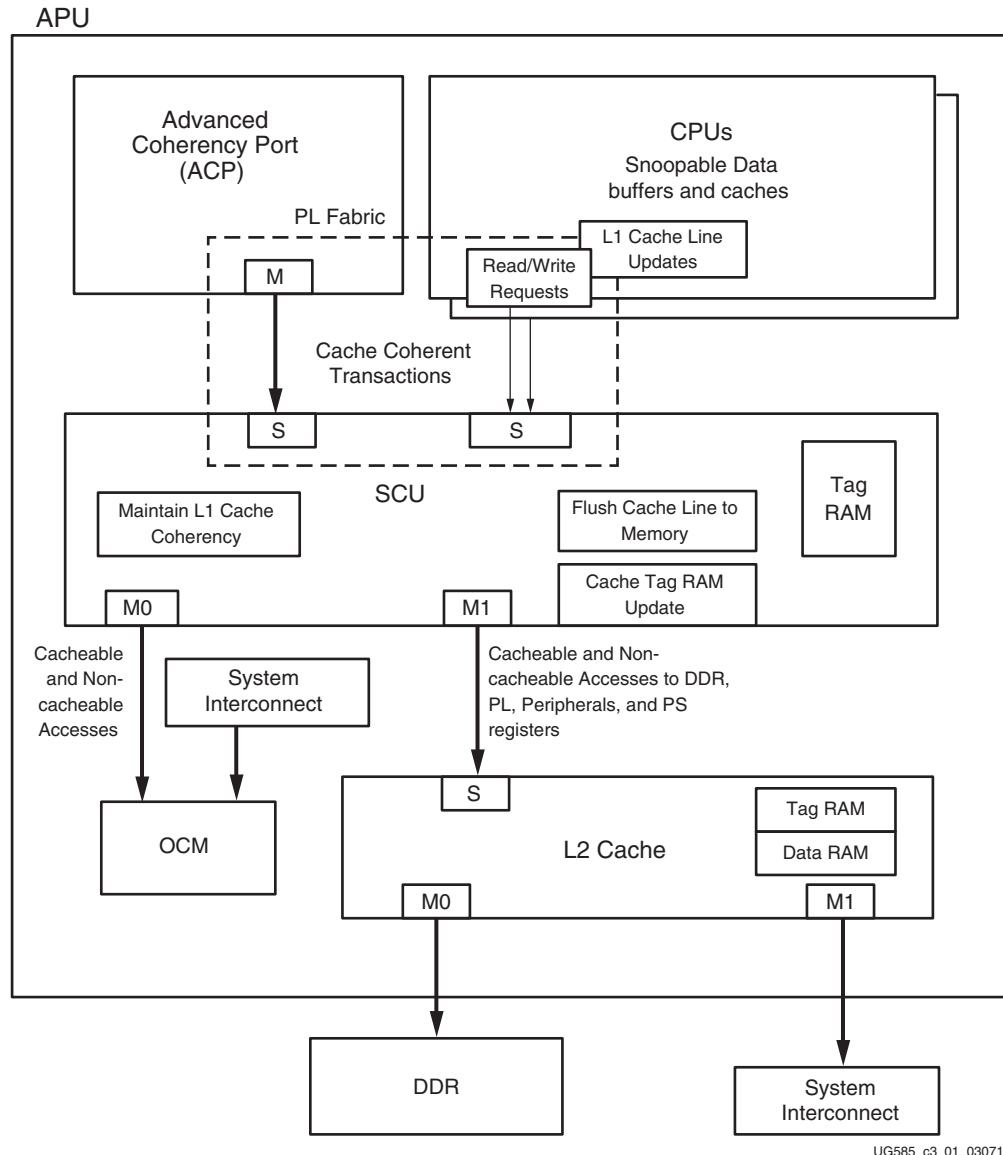


Figure 3-1: APU Block Diagram

ARM architecture supports multiple operating modes including supervisor, system, and user modes to provide different levels of protection at the application level. The architecture support for TrustZone technology helps to create a secure environment to run applications and protect their contents. TrustZone built into the ARM CPU processor and many peripherals enable a secure system to handle keys, private data, and encrypted information without allowing these secrets to leak to non-trusted programs or users.

The APU contains a 32-bit watchdog timer and a 64-bit global timer with auto-decrement features that can be used as general-purpose timers and also as a mechanism to wake up the processors from the sleep mode.

3.1.2 System-level View

The APU is the most critical component of the system that comprises the PS, the IPs implemented in the PL, and board-level devices such as the external memories and the peripherals. The main interfaces through which the APU communicates to the rest of the system are two interfaces through the L2 controller and an interface to the OCM that is parallel to the L2 cache. See [Figure 3-1](#).

All accesses from the dual Cortex-A9 MP system go through the SCU and all accesses from any other master that requires coherency with the Cortex-A9 MP system also need to be routed through the SCU using the ACP Port. All accesses that are not routed through the SCU are non-coherent with the CPU and software has to explicitly handle the synchronization and coherency.

Accesses from the APU can target the OCM, DDR, PL, IOP slaves, or registers within the PS sub-blocks. In order to minimize the latency to the OCM, a dedicated master port from the SCU provides direct access by the processors and the ACP to the OCM, offering a latency that is even less than the L2 cache.

All APU accesses to the DDR are routed through the L2 cache controller. To improve the latencies of the DDR accesses, there is a dedicated master port from the L2 cache controller to the DDR memory controller that allows all APU-DDR transactions to bypass the main interconnects which is shared with the other masters. All other accesses from the APU that are neither OCM-bound nor DDR-bound go through the L2 controller and are routed through the main interconnect using a second port. The accesses that pass through the L2 cache controller do not have to be cacheable.

As shown in [Figure 3-2](#), the APU and its sub-blocks all operate in the CPU_6x4x clock domain. The interfaces from the APU to the OCM and to the main interconnects are all synchronous. The main interconnects can run at 1/2 or 1/3 of the frequency of the CPU. The DDR block is on the DDR_3x clock domain and operates asynchronously to the APU. The ACP port to the APU block includes a synchronizer and the PL master that uses this port can have a clock that is asynchronous to the APU.

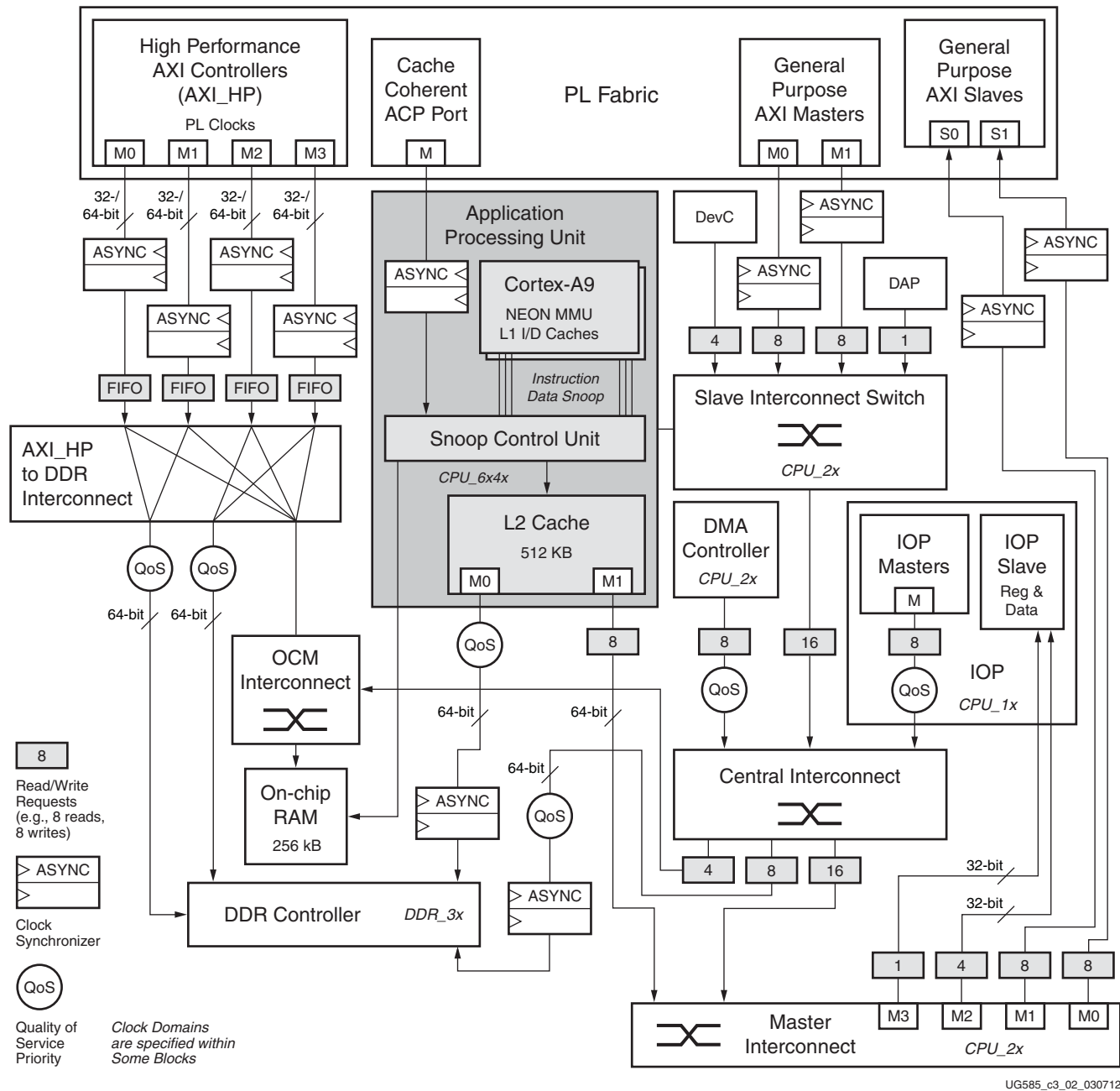


Figure 3-2: APU System View Diagram

3.2 Cortex A9 Processors

3.2.1 Summary

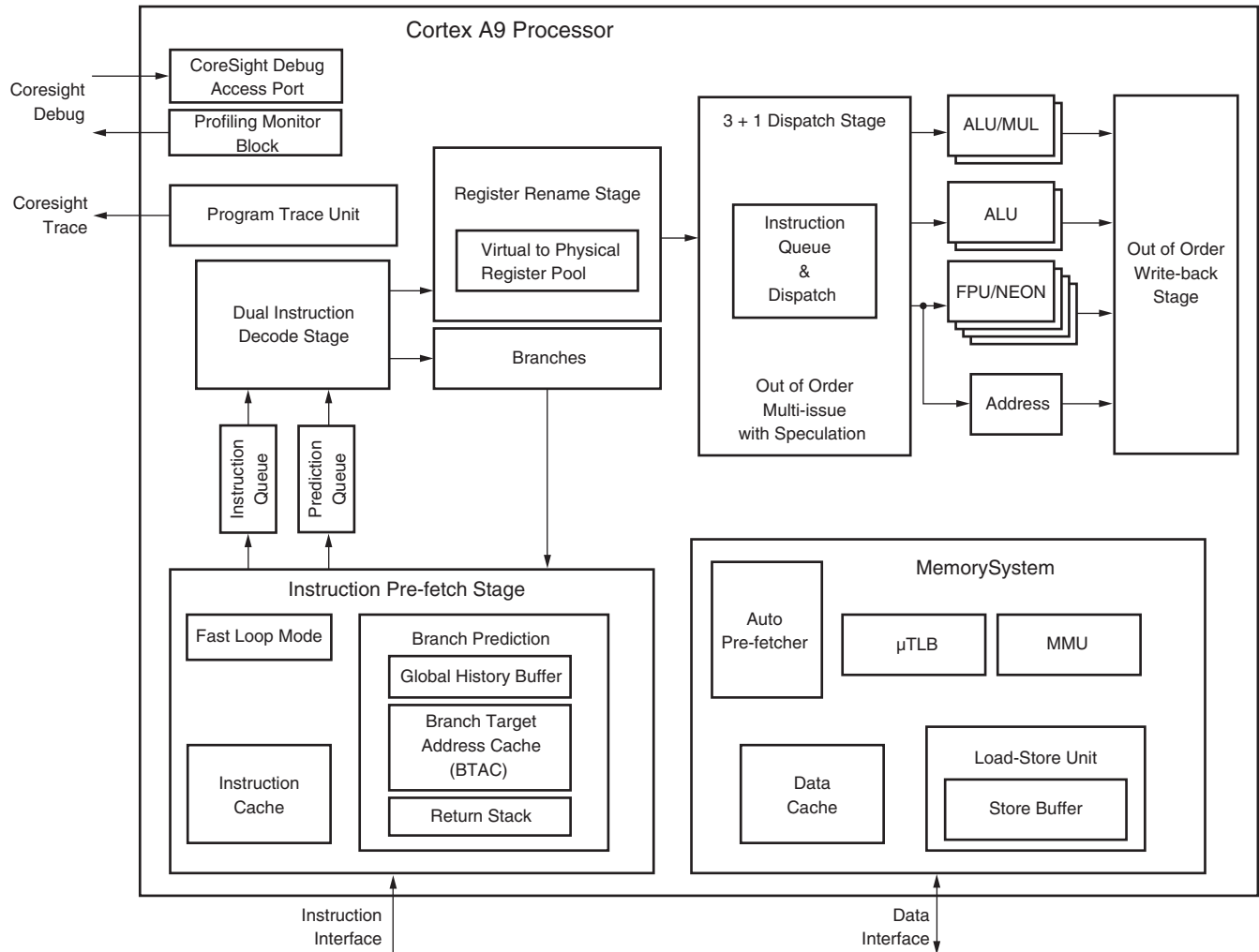
The APU implements a dual-core Cortex A9 MP configuration. Each processor has its own SIMD media processing engine (NEON), memory management unit (MMU), and separate 32 KB level-one (L1) instruction and data caches. Each Cortex-A9 processor provides two 64-bit AXI master interfaces for independent instruction and data transactions to the SCU. Depending on the address and attributes, these transactions are routed to the OCM, L2 cache, DDR memory, or, through the PS interconnect, to other slaves in the PS, or to the PL. Each processor's interface with the SCU includes the required snoop signals to provide coherency between the L1 data caches within the processors and the shared L2 cache for shareable memory. The Cortex A9 and its subsystem also provide complete Trustzone extension, necessary for user security.

The Cortex-A9 processor implements the necessary hardware features for program debug and trace generation support. The processor also provides hardware counters to gather statistics on the operation of the processor and memory system.

The major sub-blocks within the Cortex-A9 are the central processing unit (CPU), the L1 instruction and data caches, the memory management unit (MMU), the NEON coprocessor, and the core interfaces. Their functions are explained in the following subsections.

3.2.2 Central Processing Unit (CPU)

Each Cortex A9 CPU can issue two instructions in one cycle and execute them out of order. The CPU implements dynamic branch prediction and with its variable length pipeline can deliver 2.5 DMIPs/MHz. The Cortex-A9 processor implements the ARMv7-A architecture with full virtual memory support and can execute 32-bit ARM instructions, 16-bit and 32-bit Thumb instructions, and 8-bit Java™ byte codes in the Jazelle hardware acceleration state. [Figure 3-3](#) shows the architecture of the Cortex A9 processor.



UG585_c3_04_030712

Figure 3-3: Cortex A9 Architecture

Pipeline

The pipeline implemented in the Cortex-A9 CPU employs advanced fetching of instructions and branch prediction that decouples the branch resolution from potential memory latency-induced instruction stalls. In the Cortex-A9 CPU, up to four instruction-cache lines are pre-fetched to reduce the impact of memory latency on the instruction throughput. The CPU fetch unit can continuously forward two to four instructions per cycle to the instruction decode buffer to ensure efficient superscalar pipeline utilization. The CPU implements a superscalar decoder capable of decoding two full instructions per cycle, and any of the four CPU pipelines can select instructions from the issue queue. The parallel pipelines support concurrent execution across full dual arithmetic units, load-store unit, plus resolution of any branch each cycle.

The Cortex-A9 CPU employs speculative execution of instructions enabled by dynamic renaming of physical registers into an available pool of virtual registers. The CPU employs this virtual register renaming to eliminate dependencies across registers without jeopardizing the correct execution of programs. This feature allows code acceleration through an effective hardware based unrolling of

loops, and increases the pipeline utilization by removing data dependencies between adjacent instructions which also indirectly reduces interrupt latency.

In the Cortex-A9 CPU, dependent load-store instructions can be forwarded for resolution within the memory system to further reduce pipeline stalls. The core supports up to four data cache line fill requests that can be through automatic or user-driven pre-fetching.

A key feature of this CPU is the out-of-order write back of instructions that enables the pipeline resources to be released independent of the order in which the system provides the required data.

Load/store instructions can be issued speculatively before condition of instruction or a preceding branch has been resolved or before data to be written has become available. If the condition required for the execution of the load/store fails, any of the side-effects, such as the action to modify registers, are flushed.

Branch Prediction

To minimize the branch penalty in its highly pipelined CPU, the Cortex A9 implements both static and dynamic branch prediction. Static branch prediction is provided by the instructions and is decided during compilation. Dynamic branch prediction uses the outcome of the previous executions of a specific branch to determine whether the branch should be taken or not. The dynamic branch prediction logic employs a global branch history buffer (GHB) which is a 4096-entry table holding 2-bit prediction information for specific branches and is updated every time a branch gets executed.

The branch execution and the overall instruction throughput also benefit greatly from the implementation of a branch target address cache (BTAC) which holds the target addresses of the recent branches. This 512-entry address cache is organized as 2-way \times 256 entries and provides the target address for a specific branch to the pre-fetch unit before the actual target address is generated based on the calculation of the effective address and its translation to the physical address. Additionally, if an instruction loop fits in four BTAC entries, instruction cache accesses are turned off to lower power consumption.

Note: Both GHB and BTAC RAMs implement parity for protection; however, this support has limited diagnostic value. Corruption in GHB data or BTAC data does not generate functional errors in the Cortex-A 9 processor. Corruption in GHB data or BTAC data results in a branch misprediction, that is, detected and corrected when the branch gets executed.

The Cortex-A9 CPU can predict conditional branches, unconditional branches, indirect branches, PC-destination data-processing operations, and branches that switch between ARM and Thumb states. However, the following branch instructions are not predicted:

- Branches that switch between states (except ARM to Thumb transitions, and Thumb to ARM transitions).
- Instructions with the S suffix are not predicted as they are typically used to return from exceptions and have side effects that can change privilege mode and security state.
- All mode-changing instructions.

Users can enable program flow prediction by setting the Z bit in the CP15 c1 Control register to 1. Refer to the System Control Register in the *ARM Cortex-A9 Technical Reference Manual* (see [Appendix A, Additional Resources](#)). Before switching the program flow prediction on, a BTAC flush operation must be performed which has the additional effect of setting the GHB into a known state.

Cortex-A9 also employs an 8-entry return stack cache that holds the 32-bit subroutine return addresses. This feature greatly reduces the penalty of executing subroutine calls and can address nested routines up to eight levels deep.

Instruction and Data Alignment

ARM architecture specifies the ARM instructions as being 32-bits wide and requires them to be word-aligned. Thumb instructions are 16-bits wide and are required to be half-word aligned. Thumb-2 instructions which are 16- or 32-bits wide are also required to be half-word aligned. Data accesses can be unaligned and the load/store unit within the CPU breaks them up to aligned accesses. The data from these accesses are merged and sent to the register file within the CPU as had been requested.

Note: The application processing unit (APU), and the PS as a whole, support only little-endian architecture for both instruction and data.

Trace and Debug

The Cortex-A9 processor implements the ARMv7 debug architecture as described in the ARM Architecture Reference Manual. In addition, the processor supports a set of Cortex-A9 processor-specific events and system-coherency events. For more information, see *Chapter 11, Performance Monitoring Unit* in the *ARM Cortex-A9 Technical Reference Manual*.

The debug interface of the processor consists of:

- A baseline CP14 interface that implements the ARMv7 debug architecture and the set of debug events as described in the ARM Architecture Reference Manual.
- An extended CP14 interface that implements a set of debug events specific to this processor (explained in the ARM Architecture Reference Manual).
- An external debug interface connected to an external debugger through a debug access port (DAP).

The Cortex-A9 includes a program trace module that provides ARM CoreSight technology compatible program-flow trace capabilities for either of the Cortex-A9 processors and provides full visibility into the processor's actual instruction flow. The Cortex-A9 PTM includes visibility over all code branches and program flow changes with cycle-counting enabling profiling analysis. The PTM block in conjunction with the CoreSight Design Kit provides the software developer the ability to non-obtrusively trace the execution history of multiple processors and either store this, along with time stamped correlation, into an on-chip buffer, or off chip through a standard trace interface so as to have improved visibility during development and debug.

The Cortex A9 processor also implements program counters and event monitors that can be configured to gather statistics on the operation of the processor and the memory system.

3.2.3 Level 1 Caches

Each of the two Cortex-A9 processors has separate 32 KB level-1 instruction and data caches. Both L1 caches have common features that include:

- Each cache can be disabled independently, using the system control coprocessor. Refer to the System Control Register in the *ARM Cortex-A9 Technical Reference Manual*.
- The cache line lengths for both L1 caches are 32 bytes.
- Both caches are 4-way set-associative.
- L1 caches support 4 KB, 64 KB, 1 MB, and 16 MB virtual memory page.
- Neither of the two L1 caches supports the lock-down feature.
- The L1 caches have 64-bit interfaces to the integer core and AXI master ports.
- Cache replacement policy is either pseudo round-robin or pseudo random. The victim counter is read at time of miss, not allocation and it is incremented on allocation. An invalid line in the set is replaced in preference to using the victim counter.
- On a cache miss, critical word first filling of the cache is performed.
- To reduce power consumption, the number of full cache reads is reduced by taking advantage of the sequential nature of many cache operations. If a cache read is sequential to the previous cache read, and the read is within the same cache line, only the data RAM set that was previously read is accessed.
- Both L1 caches support parity.
- All memory attributes are exported to external memory systems.
- Support for TrustZone security exports the secure or non-secure status to the caches and memory system.
- Upon reset, the contents of both L1 caches are cleared to comply with security requirements

Note: The user must invalidate the instruction cache, the data cache, and BTAC before using them. It is not required to invalidate the main TLB, even though it is recommended for safety reasons. This ensures compatibility with future revisions of the processor.

The L1 instruction-side cache (I-Cache) is responsible for providing an instruction stream to the Cortex-A9 processor. The L1 I-Cache interfaces directly to the pre-fetch unit which contains a two-level prediction mechanism as described in the [Branch Prediction](#) section of this chapter. The L1 instruction cache is virtually indexed and physically tagged.

The L1 data-side cache (D-Cache) is responsible for holding the data used by the Cortex-A9 processor. Key features of the L1 D-Cache include:

- Data cache is physically indexed and physically tagged.
- D-Cache is non-blocking and therefore, load/store instructions can continue to hit the cache while it is performing allocations from external memory due to prior read/write misses. The data cache supports four outstanding reads and four outstanding writes.
- The CPU can support up to four outstanding preload (PLD) instructions. However, explicit load/store instructions have higher priority.
- The Cortex-A9 load/store unit supports speculative data pre-fetching which monitors sequential accesses made by program and starts fetching the next expected line before it has been requested. This feature is enabled in the cp15 Auxiliary Control register (DP bit). The pre-fetched lines can be dropped before allocation, and PLD instruction has higher priority.
- The data cache supports two 32-byte line-fill buffers and one 32-byte eviction buffer.
- The Cortex-A9 CPU has a store buffer with four 64-bit slots with data merging capability.

- Both data cache read misses and write misses are non-blocking with up to four outstanding data cache read misses and up to four outstanding data cache write misses being supported.
- The APU data caches offer full snoop coherency control utilizing the MESI algorithm.
- The data cache in Cortex-A9 contains local load/store exclusive monitor for LDREX/STREX synchronizations. These instructions are used to implement semaphores. The exclusive monitor handles one address only, with eight 8 words or one cache line granularity. Therefore, avoid interleaving LDREX/STREX sequences and always execute a CLREX instruction as part of any context switch.
- D-Cache only supports write-back/write-allocate policy. Write-through and write-back/no write-allocate policies are not implemented.
- L1 D-Cache offers support for exclusive operation with respect to the L2 cache. Exclusive operation implies that a cache line is valid only in L1 or L2 cache and never in both at the same time. A line-fill into L1 causes the line to be marked invalid in L2. At the same time, eviction of a line from L1 causes the line to be allocated in L2, even if it is not dirty. A line-fill into L1 from dirty L2 line forces eviction of the line to external memory. The exclusive operation, disabled by default, increases cache utilization and reduces power consumption.

3.2.4 Memory Management Unit (MMU)

The MMU in the ARM architecture involves both memory protection and address translation. The MMU works closely with the L1 and L2 memory systems in the process of translating virtual addresses to physical addresses. It also controls accesses to and from the external memory.

The MMU is compatible with the Virtual Memory System Architecture version 7 (VMSAv7) requirements supporting 4 KB, 64 KB, 1 MB, and 16 MB page table entries and 16 access domains. The unit provides global and application-specific identifiers to remove the requirement for context switch TLB flushes and has the capability for extended permission checks. Please see the ARM Architecture Reference Manual for a full architectural description of the VMSAv7.

The processor implements the ARMv7-A MMU enhanced with security extensions and multiprocessor extensions to provide address translation and access permission checks. The MMU controls table-walk hardware that accesses translation tables in main memory. The MMU enables fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes held in instruction and data translation look-aside buffers (TLBs).

In summary, the MMU is responsible for the following operations:

- Checking of Virtual Address and ASID (address space identifier)
- Checking of domain access permissions
- Checking of memory attributes
- Virtual-to-physical address translation
- Support for four page (region) sizes
- Mapping of accesses to cache, or external memory
- Four entries in the main TLB are lockable

Memory Access Sequence

When the processor generates a memory access, the MMU:

1. Performs a look-up for the requested virtual address and current ASID and security state in the relevant instruction or data micro TLB.
2. If there is a miss in the micro TLB, performs a look-up for the requested virtual address and current ASID and security state in the main TLB.
3. If there is a miss in main TLB, performs a hardware translation table walk.

The MMU might not find a global mapping or a mapping for the currently selected ASID with a matching non-secure TLB ID (NSTID) for the virtual address in the TLB. In this case, the hardware does a translation table walk if the translation table walk is enabled by the PD0 or PD1 bit in the TTB Control register. If translation table walks are disabled, the processor returns a section translation fault.

If the MMU finds a matching TLB entry, it uses the information in the entry as follows:

1. The access permission bits and the domain determine if the access is enabled. If the matching entry does not pass the permission checks, the MMU signals a memory abort. See the ARM Architecture Reference Manual for a description of access permission bits, abort types and priorities, and for a description of the Instruction Fault Status register (IFSR) and Data Fault Status register (DFSR).
2. The memory region attributes specified in both the TLB entry and the CP15 c10 remap registers control the cache and write buffer, and determine if the access is:
 - a. Secure or non-secure
 - b. Shared or not
 - c. Normal memory, device, or strongly-ordered
3. The MMU translates the virtual address to a physical address for the memory access.

If the MMU does not find a matching entry, a hardware table walk occurs.

TLB Organization

The Cortex-A9 MMU includes two levels of TLBs which include a unified TLB for both instruction and data and separate micro TLBs for each. The micro TLBs act as the first level TLBs and each have 32 fully associative entries. If an instruction fetch or a load/store address misses in the corresponding micro TLB, the unified or main TLB is accessed. The unified main TLB provides a 2-way associative 2x64 entry table (128 entries) and supports 4 lockable entries using the lock-by-entry model. The TLB uses a pseudo round-robin replacement policy to determine which entry in the TLB should be replaced in the case of a miss.

Unlike some other risk processors that require software to manage the updates of the TLB from the page table that resides in the memory, the main TLB in Cortex-A9 supports hardware page table walks to perform look-ups in the L1 data cache. This allows the page tables to be cached.

The MMU can be configured to perform hardware translation table walks in cacheable regions by setting the IRGN bits in the Translation Table Base registers. If the encoding of the IRGN bits is write-back, then an L1 data cache look-up is performed and data is read from the data cache. If the

encoding of the IRGN bits is write-through or non-cacheable, then an access to external memory is performed.

TLB entries can be global, or can be assigned to particular processes or applications using the ASIDs associated with those processes. ASIDs enable TLB entries remain resident during context switches, avoiding the requirement of reloading them subsequently.

Note: The ARM Linux kernel manages the 8-bit TLB ASID space globally across all CPUs instead of on a per-CPU basis. The ASID is incremented for each new process. When the ASID rolls over (ASID = 0,) a TLB flush request is sent to both CPUs. However, only the CPU that is in the middle of a context switch immediately updates its current ASID context. The other CPU continues to run using its current pre-rollover ASID until a scheduling interval occurs and then it context switches to a new process.

TLB maintenance and configuration operations are controlled through a dedicated coprocessor, CP15, integrated within the core. This coprocessor provides a standard mechanism for configuring the level one memory system.

Micro TLB

The first level of caching for the page table information is a micro TLB of 32 entries implemented on each of the instruction and data sides. These blocks provide a fully associative look-up of the virtual addresses in a cycle.

The micro TLB returns the physical address to the cache for the address comparison, and also checks the protection attributes to signal either a pre-fetch abort or a data abort.

All main TLB related operations affect both the instruction and data micro TLBs, causing them to be flushed. In the same way, any change of the Context ID register causes the micro TLBs to be flushed. The main or unified TLB, explained in the next section, should be invalidated after reset and before the MMU is enabled.

Main TLB

The main TLB is the second layer in the TLB structure that catches the misses from the Micro TLBs. It also provides a centralized source for lockable translation entries.

Misses from the instruction and data micro TLBs are handled by a unified main TLB. Accesses to the main TLB take a variable number of cycles, according to competing requests from each of the micro TLBs and other implementation-dependent factors.

Entries in the lockable region of the main TLB are lockable at the granularity of a single entry. As long as the lockable region does not contain any locked entries, it can be allocated with non-locked entries to increase overall main TLB storage size.

TLB Match Process

Each TLB entry contains a virtual address, a page size, a physical address, and a set of memory properties. Each is marked as being associated with a particular application space, or as global for all application spaces. A TLB entry matches if bits [31: N] of the modified virtual address (MVA) match, where N is \log_2 of the page size for the TLB entry. It is either marked as global, or the ASID matched the current ASID.

A TLB entry matches when these conditions are true:

- Its virtual address matches that of the requested address
- Its non-secure TLB ID (NSTID) matches the secure or non-secure state of the MMU request
- Its ASID matches the current ASID or is global

The operating system must ensure that, at most, one TLB entry matches at any time. A TLB can store entries based on the following block sizes:

Supersections: 16 MB blocks of memory

Sections: 1 MB blocks of memory

Large pages: 64 KB blocks of memory

Small pages: 4 KB blocks of memory

Supersections, sections, and large pages are supported to permit mapping of a large region of memory while using only a single entry in a TLB. If no mapping for an address is found within the TLB, then the translation table is automatically read by hardware and a mapping is placed in the TLB.

TLB Lockdown

The TLB supports the TLB lock-by-entry model as described in the ARM Architecture Reference Manual. See TLB Lockdown Register description in the ARM Cortex-A9 Technical Reference Manual.

3.2.5 Interfaces

AXI and Coherency Interfaces

Each Cortex-A9 processor provides two 64-bit pseudo AXI master interfaces for independent instruction fetch and data transactions. These interfaces operate at the speed of the processor cores (CPU_6x4x clock) and are capable of sustaining four double-word writes every five processor cycles when copying data across a cached region of memory. The instruction side interface is a read-only interface and does not have the write channel. These interfaces implement an extended version of the AXI protocol that also provides multiple optimizations to the L2 cache including support for L2 pre-fetch hints and speculative memory accesses. These optimizations are explained in more detail in the [L2-Cache](#) section of this chapter.

The AXI transactions are all routed through the SCU to the OCM or the L2 cache controller based on their addresses. Each Cortex-A9 also provides a cache coherency bus (CCB) to the SCU to provide the information required for coherency management between the L1 and L2 caches.

Debug and Trace Interfaces

Each Cortex-A9 processor has a standard 32-bit APB slave port that operates at the CPU_1x clock frequency and is accessed through the debug APB bus master in the SOC debug block. The operation of this block is explained in the corresponding chapter of this document.

The Cortex-A9 processors also include a pair of interfaces for trace generation and cross trigger control. The trace source interface from each core is a 32-bit CoreSight standard ATB master port that operates at the speed of the PS interconnect (CPU_2x clock), and is connected to the funnel in the SOC debug block. Each core also has a 4-bit standard CoreSight cross trigger interface that operates at the interconnect frequency (CPU_2x clock) and is connected to the cross trigger matrix (CTM) in the SOC debug block.

Other Interfaces

Each Cortex-A9 processor has multiple control bits that are driven through the System-Level Control register (SLCR). This includes a 4-bit interface that drives the CoreSight standard security signals and also static configuration signals for controlling CP15 and SW programmability.

There are also other interfaces including the event and interrupt interfaces that are explained later in this chapter.

3.2.6 NEON

- The Cortex-A9 NEON MPE extends the Cortex-A9 functionality to provide support for the ARM v7 advanced SIMD and vector floating-point v3 (VFPv3) instruction sets. The Cortex-A9 NEON MPE supports all addressing modes and data-processing operations described in the ARM Architecture Reference Manual.

The Cortex-A9 NEON MPE features are:

- SIMD vector and scalar single-precision floating-point computation
 - Unsigned and signed integers
 - Single bit coefficient polynomials
 - Single-precision floating-point values
- The operations supported by the NEON co-processor include:
 - Addition and subtraction
 - Multiplication with optional accumulation
 - Maximum or minimum value driven lane selection operations
 - Inverse square-root approximation
 - Comprehensive data-structure load instructions, including register-bank-resident table lookup.
- Scalar double-precision floating-point computation
- SIMD and scalar half-precision floating-point conversion
- 8, 16, 32, and 64-bit signed and unsigned integer SIMD computation
- 8 or 16-bit polynomial computation for single-bit coefficients
- Structured data load capabilities
- Dual issue with Cortex-A9 processor ARM or Thumb instructions
- Independent pipelines for VFPv3 and advanced SIMD instructions

- Large, shared register file, addressable as:
 - Thirty-two 32-bit S (single) registers
 - Thirty-two 64-bit D (double) registers
 - Sixteen 128-bit Q (quad) registers

See the ARM Architecture Reference Manual for details of the advanced SIMD instructions and the NEON MPE operation.

3.2.7 Performance Monitoring Unit

The Cortex-A9 processor includes a performance monitoring unit (PMU) which provides six counters to gather statistics on the operation of the processor and memory system. Each counter can count any of 58 events available in the Cortex-A9 processor. The PMU counters, and their associated control registers, are accessible from the internal CP15 interface as well as from the DAP interface. For details, refer to the Performance Monitoring Unit in the *ARM Cortex-A9 Technical Reference Manual*.

3.3 Snoop Control Unit (SCU)

3.3.1 Summary

The SCU block connects the two Cortex A9 processors to the memory subsystem and contains the intelligence to manage the data cache coherency between the two processors and the L2 cache. This block is responsible for managing the interconnect arbitration, communication, cache and system memory transfers, and cache coherence for the Cortex A9 processors. The APU also exposes the capabilities of the SCU to system accelerators that are implemented in the PL through the accelerator coherence port (ACP) interface that is explained later. This interface allows PL masters to share and access the processor's cache hierarchy. The offered system coherence here not only improves performance but also reduces the software complexity involved in otherwise maintaining software coherency within each OS driver.

The SCU block communicates with each of the Cortex-A9 processors through a cache coherency bus (CCB) and manages the coherency between the L1 and the L2 caches. The SCU supports MESI snooping which provides increased power efficiency and performance by avoiding unnecessary system accesses. The block implements duplicated 4-way associative tag RAMs acting as a local directory that lists coherent cache lines held in the CPU L1 data caches. The directory allows the SCU to check if data is in the L1 data caches with great speed and without interrupting the processors. Also, accesses can be filtered only to the processor that is sharing the data.

The SCU can also copy clean data from one processor cache to another and eliminate the need for main memory accesses to perform this task. Furthermore, it can move dirty data between the processors, skipping the shared state and avoiding the latency associated with the write-back.

It is important to note that the Cortex A9 does not guarantee coherency between the L1 instructions caches as the processor is not capable of modifying the L1 contents directly.

3.3.2 Address Filtering

One of the functions of the SCU is to filter transactions that are generated by the processors and the ACP based on their addresses and route them accordingly to the OCM or L2 controller. The granularity of the address filtering within the SCU is 1 MB; therefore, all accesses by the processors or through the ACP whose addresses are within a 1 MB window can only target the OCM or L2 controller. The default setting of the address filtering within the SCU routes all the upper and lower 1M addresses within the 4G address space to the OCM and the rest of the addresses are routed to the L2 controller. Refer to the [SCU Address Filtering](#) section of [Chapter 29, On-Chip Memory \(OCM\)](#) for more information on the SCU address filtering.

3.3.3 SCU Master Ports

Each of the SCU AXI master ports to the L2 or OCM has the following write and read issuing capabilities:

- Write issuing capability:
 - 10 write transactions per processor:
 - 8 non-cacheable writes
 - 2 evictions from L1
 - 2 additional writes for eviction traffic from the SCU
 - 3 more write transactions from the ACP
- Read issuing capability:
 - 14 read transactions per processor:
 - 4 instruction reads
 - 6 linefill reads
 - 4 non-cacheable read
 - 7 more read transactions from the ACP

3.4 L2-Cache

3.4.1 Summary

The L2 cache controller is based on the ARM PL310 and includes an 8-way set-associative 512 KB cache for dual Cortex-A9 cores. The L2 cache is physically addressed and physically tagged and supports a fixed 32-byte line size. These are the main features of the L2 cache:

- Supports snoop coherency control utilizing MESI algorithm
- Offers parity check for L2 cache memory
- Supports speculative read operations in the SMP mode

- Provides L1/L2 exclusive mode (i.e., data exists in either, but not both)
- Can be locked down by master, line, or way per master
- Implements 16-entry deep preload engine for loading data into L2 cache memory
- To improve latency, critical-word-first line-fill is supported
- Implements pseudo-random victim selection policy with deterministic option
 - Write-through and write-back
 - Read allocate, write allocate, read and write allocate
- The contents of the L2 data and tag RAMS are cleared upon reset to comply with security requirements
- The L2 controller implements multiple 256-bit line buffers to improve cache efficiency
 - Line fill buffers (LFBs) for external memory access to create a complete cache line into L2 cache memory. Four LFBs are implemented for AXI read interleaving support
 - Two 256-bit line read buffers for each slave port. These buffers hold a line from the L2 cache in case of cache hit
 - Three 256-bit eviction buffers hold evicted lines from the L2 cache, to be written back to main memory
 - Three 256-bit store buffers hold bufferable writes before their draining to main memory, or L2 cache. They enable multiple writes to the same line to be merged
- The controller implements selectable cache pre-fetching within 4k boundaries.
- The L2 cache controller forwards exclusive requests from L1 to DDR, OCM, or external memory

Note: The SCU does not maintain coherency between instruction and data L1 caches, so this coherency must be maintained by software.

The L2 Cache implements TrustZone security extension to offer enhanced OS security. The non-secure (NS) tag bit is added in tag RAM and is used for lookup in the same way as an address bit. The NS-tag bit is also added in all of the buffers. The NS bit in tag RAM is used to determine the security level of evictions to DDR and OCM. The controller restricts non-secure accesses for control, configuration, and maintenance registers to restrict access to secure data.

Cache Response

This section describes the general behavior of the cache controller depending on the Cortex A9 transactions. These are the descriptions for the different type of transactions:

Bufferable	The transaction can be delayed by the interconnect or any of its components for an arbitrary number of cycles before reaching its final destination. This is usually only relevant to writes.
Cacheable	The transaction at the final destination does not have to present the characteristics of the original transaction. For writes this means that a number of different writes can be merged together. For reads this means that a location can be pre-fetched or can be fetched just once for multiple read transactions. To determine if a transaction should be cached, this attribute should be used in conjunction with the read allocate and write allocate attributes.

Read Allocate	If the transfer is a read and it misses in the cache then it should be allocated. This attribute is not valid if the transfer is not cacheable.
Write Allocate	If the transfer is a write and it misses in the cache then it should be allocated. This attribute is not valid if the transfer is not cacheable.

In the ARM architecture, the inner attributes are used to control the behavior of the L1 caches and write buffers. The outer attributes are exported to the L2 or an external memory system.

In the Cortex A9 processing system (similar to most modern processors), to improve performance and power, many optimizations are performed at many levels of the system which cannot be completely hidden from the outside world and might cause the violation of the expected sequential execution model. Examples of these optimizations are:

- Multi-issue speculative and out-of-order execution
- Usage of load/store merging to minimize the latency of load/stores
- In a multi-core processor, hardware-based cache coherency management can cause cache lines to migrate transparently between cores causing different cores to see updates to cached memory locations in different orders
- External system characteristics might create additional challenges when external masters are included in the coherent system through the ACP

Therefore, it is vital to define certain rules to constrain the order in which the memory accesses of one core relate to the surrounding instructions, or could be observed by other cores within a multi-core processor system. Typically the memory is categorized as:

Normal	This memory type is typically used for all data and executable code, and permits speculative reads, merging of accesses and (if interrupted by an exception) repeating of writes without side effects. Accesses to normal memory can always be buffered, and in most situations they are also cached, but they can be configured to be un-cached. There is no implicit ordering of normal memory accesses, beyond pure address dependencies and control dependencies.
Strongly-Ordered	This memory type is typically used with memory-mapped peripherals or control registers. Accesses to these types of memory must occur exactly as the number of times that the program specifies. There is however, no guaranteed ordering between memory accesses to different devices, or between accesses of different memory types.
Device	This memory type is similar to the strongly-ordered memory type except that the device memory could be shareable or non-shareable.

Table 3-1 shows the general behavior of the L2 cache controller in response to ARMv7 load/store transaction types that are supported by Cortex-A9.

Table 3-1: Cache Controller Behavior for SCU Requests

Transaction Type	ARMv7 Equivalent	L2 Cache Controller Behavior
Non-cacheable and non-bufferable	Strongly ordered	<ul style="list-style-type: none"> Read: Not cached in L2, results in memory access. Write: Not buffered, results in memory access.
Bufferable only	Device	<ul style="list-style-type: none"> Read: Not cached in L2, results in memory access. Write: Placed in store buffer, not merged, immediately drained to memory.
Cacheable but do not allocate	Outer non-cacheable	<ul style="list-style-type: none"> Read: Not cached in L2, results in memory access. Write: Placed in store buffer, write to memory when store buffer is drained.
Cacheable write-through, allocate on read	Outer write-through, no write allocate	<ul style="list-style-type: none"> Read hit: Read from L2. Read miss: Line fill to L2. Write hit: Put in store buffer, write to L2 and memory when store buffer is drained. Write miss: Put in store buffer, write to memory when store buffer is drained.
Cacheable write-back, allocate on read	Outer write-back, no write allocate	<ul style="list-style-type: none"> Read hit: Read from L2. Read miss: Line fill to L2. Write hit: Put in store buffer, write to L2 when store buffer is drained and mark line as dirty. Write miss: Put in store buffer, write to L3 when store buffer is drained.
Cacheable write-through, allocate on write	-	<ul style="list-style-type: none"> Read hit: Read from L2. Read miss: Not cached in L2, causes memory access. Write hit: Put in store buffer, write to L2 and memory when store buffer is drained. Write miss: Put in store buffer. When buffer is drained, check if it is full. If not full, request word or line to memory before allocating buffer to L2. Allocation to L2. Write to memory.
Cacheable write-back, allocate on write	-	<ul style="list-style-type: none"> Read hit: Read from L2. Read miss: Not cached in L2, causes memory access. Write hit: Put in store buffer, write to L2 when store buffer is drained, and mark line as dirty. Write miss: Put in store buffer. When buffer has to be drained, check if it is full. If it is not full then request word or line to memory before allocating the buffer to L2. Allocation to L2.

Table 3-1: Cache Controller Behavior for SCU Requests (Cont'd)

Transaction Type	ARMv7 Equivalent	L2 Cache Controller Behavior
Cacheable write-through, allocate on read and write	Outer write-through, allocate on both reads and writes	<ul style="list-style-type: none"> Read hit: Read from L2. Read miss: Line fill to L2. Write hit: Put in store buffer, write to L2 and memory when store buffer is drained. Write miss: Put in store buffer. When buffer has to be drained, check whether it is full. If it is not full then request word or line to memory before allocating the buffer to the L2. Allocation to L2. Write to memory.
Cacheable write-back, allocate on read and write	Outer write-back, write allocate	<ul style="list-style-type: none"> Read hit: Read from L2. Read miss: Line fill to L2. Write hit: Put in store buffer, write to L2 when store buffer is drained, and mark line as dirty. Write miss: Put in store buffer. When buffer has to be drained, check if it is full. If it is not full then request word or line to memory before allocating the buffer to L2. Allocation to L2.

3.4.2 Exclusive L2-L1 Cache Configuration

In the exclusive cache configuration mode, the L1 data cache of the Cortex-A9 processor and the L2 cache are exclusive. At any time, a given address is cached in either L1 data cache or in the L2 cache, but not in both. This has the effect of increasing the usable space and efficiency of the L2 cache. When exclusive cache configuration is selected:

- Data cache line replacement policy is modified so that the victim line in the L1 always gets evicted to the L2, even if it is clean.
- If a line is dirty in the L2 cache, a read request to this address from the processor causes write-back to external memory and a line-fill to the processor.

Both L1 and L2 caches have to be configured for exclusive caching. Setting the exclusive cache configuration bit 12 in the Auxiliary Control register for L2 and bit 7 of the ACTLR register in Cortex-A9 configure the L2 and L1 caches to operate exclusive to one another.

For reads, the behavior is as follows:

- For a hit, the line is marked as non-valid (the tag RAM valid bit is reset,) and the dirty bit is unchanged. If the dirty bit is set, future accesses can still hit in this cache line but the line is part of the preferred choice for future evictions.
- For a miss, the line is not allocated into the L2 cache.

For writes, the behavior depends on the value of attributes from the SCU to indicate if the write transaction is an eviction from the L1 memory system and whether it is a clean eviction. AWUSERS[8] attribute indicates an eviction and AWUSERS[9] indicates a clean eviction. The behavior is summarized as follows:

- For a hit, the line is marked dirty unless the AWUSERS[9:8] = b11. In this case, the dirty bit is unchanged.

- For a miss, if the cache line is evicted (AWUSERS[8] is 1,) the cache line is allocated and its dirty status depends on if it is evicted dirty or not. If the cache line is evicted dirty (AWUSERS[8] is 0,) the cache line is allocated only if it is write allocate.

3.4.3 Cache Replacement Strategy

Bit [25] of the Auxiliary Control register configures the replacement strategy. It can be either round-robin or pseudo-random. The round-robin replacement strategy fills invalid and unlocked ways first; for each line, when ways are all valid or locked, the victim is chosen as the next unlocked way. The pseudo-random replacement strategy fills invalid and unlocked ways first; for each line, when ways are all valid or locked, the victim is chosen randomly between unlocked ways.

When a deterministic replacement strategy is required, the lockdown registers are used to prevent ways from being allocated. For example, since L2 cache is 512 KB and is 8-way set-associative, each way is 64 KB. If a piece of code is required to reside in two ways (128 KB,) with a deterministic replacement strategy, ways 1-7 must be locked before the code is filled into the L2 cache. If the first 64 KB of code is allocated into way 0 only, then way 0 must be locked and way 1 unlocked so that the second half of the code can be allocated in way 1.

There are two lockdown registers, one for data and one for instructions. If required, one can separate data and instructions into separate ways of the L2 cache.

3.4.4 Cache Lockdown

The L2 cache controller allows locking down entries by line, by way, or by master (includes both CPU and ACP masters.) Lockdown by line and lockdown by way can be used at the same time; lockdown by line and lockdown by master can also be used at the same time. However, lockdown by master and lockdown by way are exclusive, because lockdown by way is a subset of lockdown by master.

Lockdown by Line

When enabled, all newly allocated cache lines get marked as locked. The controller then considers them as locked and does not naturally evict them. It is enabled by setting bit [0] of the lockdown by the Line Enable register. Bit [21] of the tag RAM shows the locked status of each cache line.

Note: An example of when the lockdown by line feature might be enabled is during the time when a critical piece of software code is loaded into the L2 cache.

The unlock all lines background operation enables the unlocking of all lines marked as locked by the lockdown by line mechanism. The status of this operation can be checked by reading the Unlock All Lines register. While an unlock all lines operation is in progress, the user cannot launch a background cache maintenance operation. If attempted, a SLVERR error is returned.

Lockdown by Way

The L2 cache is 8-way set-associative and allows users to lock the replacement algorithm on a way basis, enabling the set count to be reduced from 8-way all the way down to direct mapped. The 32-bit cache address consists of the following fields:

[Tag Field], [Index Field], [Word Field], [Byte Field]

When a cache lookup occurs, the index defines where to look in the cache ways. The number of ways defines the number of locations with the same index referred to as a set. Therefore, an 8-way set associative cache has eight locations where an address with Index A can exist. There are 2^{11} or 2,024 indices in the 512K L2 cache.

Lockdown format C, as the ARM Architecture Reference Manual describes, provides a method to restrict the replacement algorithm used for allocations of cache lines within a set. This method enables:

- Fetch of code or load data into the L2 cache
- Protection from being evicted because of other accesses
- This method can also be used to reduce cache pollution

The Lockdown register in the L2 cache controller is used to lock any of the eight ways in the L2 cache. To apply lockdown, the user sets each bit to 1 to lock each respective way. For example, set Bit [0] for Way 0, Bit [1] for Way 1.

Lockdown by Master

The lockdown by master feature is a superset of the lockdown by way feature. It enables multiple masters to share the L2 cache and makes the L2 cache behave as though these masters have dedicated smaller L2 caches. This feature enables the user to reserve ways of the L2 cache to specific master IDs.

The L2 cache controller lockdown by master is only able to distinguish up to eight different masters. However, there are up to 64 AXI master IDs from the Cortex-A9 MP core. Table 3-2 shows how the 64 master ID values are grouped into eight lockable groups.

Table 3-2: Lockdown by Master ID Group

ID Group	Comment
A9 Core 0	All 8 read/write request from Core 0
A9 Core 1	All 8 read/write request from Core 1
A9 Core 2	Reserved for future
A9 Core 3	Reserved for future
ACP Group0	ACP ID = {000, 001}
ACP Group1	ACP ID = {010, 011}
ACP Group2	ACP ID = {100, 101}
ACP Group03	ACP ID = {110, 111}

3.4.5 Enabling and Disabling the L2 Cache Controller

The L2 cache is disabled by default and can be enabled by setting Bit 0 of the L2 Cache Control register independently of the L1 caches. When the cache controller block is not enabled, depending on their addresses, transactions pass through to the DDR memory or the main interconnect on the

cache controller master ports. The address latency introduced by the disabled cache controller is one cycle in the slave port from the SCU plus one cycle in the master ports.

3.4.6 RAM Access Latency Control

The L2 cache data and tag RAMs use the same clock as the Cortex-A9 processors; however, it is not feasible to access these RAMs in a single cycle when the clock runs at its maximum speed. To address this issue, the L2-cache controller provides a mechanism to adjust the latencies for the write access, read access, and setup of both RAM arrays by respectively setting bits [10:8], [6:4], and [2:0] of its Tag RAM and Data RAM Latency Control registers. The default value for these fields is 3'b111 for both registers which corresponds to the maximum latency of eight CPU_6x4x cycles for the three attributes of each RAM array. Because these large latencies result in very poor cache performance, the software should reset the attributes as follows:

- Set the latencies for the three tag RAM attributes to 2 by writing 3'b001 to bits [10:8], [6:4], and [2:0] of the Tag RAM Latency Control register.
- Set the latencies for the write access and setup of the data RAM to 2 by writing 3'b001 to bits [10:8] and [2:0] of the Data RAM Latency Control register.
- Set the read access latency of the data RAM to 3 by writing 3'b010 to bits [6:4] of the Data RAM Latency Control register.

3.4.7 Store Buffer Operation

Two buffered write accesses to the same address and the same security bit cause the first write access to be overridden if the controller does not drain the store buffer after the first access. The store buffer has merging capabilities, so it merges successive writes to the same line address into the same buffer slot. This means that the controller does not drain the slots as soon as they contain data, but rather waits for other potential accesses that target the same cache line. The store buffer draining policy is as follows. Slave port refers to the port from the SCU to the L2 cache controller:

- Store buffer slot is immediately drained if targeting device memory area
- Store buffer slots are drained as soon as they are full
- Store buffer is drained at each strongly-ordered read occurrence in the slave port
- Store buffer is drained at each strongly ordered write occurrence in the slave port
- If the three slots of the store buffer contain data, the least recently accessed slot is drained
- If a hazard is detected with one store buffer slot, it is drained to resolve the hazard. Hazards can occur when data is present in the cache buffers, but not yet present in the cache RAM or external memory
- Store buffer slots are drained when a locked transaction is received by the slave port
- Store buffer slots are drained when a transaction targeting the configuration registers is received by the slave port

Merging condition is based on address and security attribute. Merging takes place only when data is in the store buffer and it is not draining.

When a write-allocate cacheable slot is drained, misses in the cache, and is not full, the store buffer sends a request via the master ports to the main interconnects or DDR to complete the cache line. The corresponding master port sends a read request through the interconnects and provides data to the store buffer in return. When the slot is full, it can be allocated into the cache.

3.4.8 Optimizations Between Cortex-A9 and L2 Controller

To improve performance, the SCU interface to the L2 controller and partially the interface to the on-chip memory controller (OCM), implement several optimizations:

- Early write response
- Pre-fetch hints
- Full line of zero write
- Speculative reads of the Cortex-A9 MPCore processor

These optimizations apply to the transfers from the processor and do not include the ACP.

Early Write Response

During the write transaction from the Cortex-A9 to the L2 cache controller, the write response from the L2 controller is normally returned to the SCU only when the last data beat has arrived at the L2 controller. This optimization enables the L2 controller to send the write response of certain write transactions as soon as the store buffer accepts the write address and allows the Cortex-A9 processor to provide a higher bandwidth for writes. This feature is disabled by default and the user can enable it by setting the Early BRESP enable bit in the Auxiliary Control register for the L2 controller. The Cortex-A9 does not require any programming to enable this feature. OCM does not support this feature and its write responses are generated normally.

Pre-fetch Hints

When the Cortex-A9 processor is configured to run in SMP mode, the automatic data pre-fetchers implemented in the CPUs issue special read accesses to the L2 cache controller. These special reads are called pre-fetch hints. When the L2 controller receives such pre-fetch hints, it allocates the targeted cache line into the L2 cache for a miss without returning any data back to the Cortex-A9 processor. The user can enable the pre-fetch hint generation by the Cortex-A9 processors through one of the two following methods:

1. Enabling the L2 pre-fetch hint feature by setting bit [1] of the ACTLR register. When enabled, this feature sets the Cortex-A9 processor to automatically issue L2 pre-fetch hint requests when it detects regular fetch patterns on a coherent memory.
2. Use of PLE (pre-load engine) operations. When this feature is used in the Cortex-A9 processor, the PLE issues a series of L2 pre-fetch hint requests at the programmed addresses.

No additional programming of the L2 Controller is required. Application of the pre-fetch hints to the OCM memory space does not cause any action because, unlike caches, transfer of data into OCM RAM requires explicit operations by software.

Full Line of Zero Write

When this feature is enabled, the Cortex-A9 processor can write entire non-coherent cache lines of zeroes to the L2 cache, using a single write command cycle. This provides a performance improvement as well as some power savings. The Cortex-A9 processor is likely to use this feature when a CPU is executing a memset routine to initialize a particular memory area.

This feature is disabled by default and can be enabled by setting the Full Line of Zero enable bit of the Auxiliary Control register for the L2 controller and the enable bit in the Cortex-A9 ACTLR register. Care must be taken if this feature is enabled because correct behavior relies on consistent enabling in both the Cortex-A9 processor and the controller.

To enable this feature, the following steps must be performed:

1. Enable Full line of zero feature in the L2 controller
2. Enable L2 cache controller
3. Enable Full line of zero feature in A9

The cache controller does not support strongly-ordered write accesses with this feature. The feature is also supported by the OCM if it is enabled in the Cortex-A9

Speculative Reads of the Cortex-A9

This is a feature unique to the Cortex-A9 MP configuration and can be enabled using a dedicated software control bit in the SCU Control register. For this feature, the Cortex-A9 has to be in the SMP mode through the use of the SMP bit in the ACTLR register; however, the L2 controller does not require any specific settings. When the speculative read feature is enabled, on coherent line fills, the SCU speculatively issues read transactions to the controller in parallel with its tag lookup. The controller does not return data on these speculative reads and only prepares data in its line read buffers.

If the SCU misses, it issues a confirmation line fill to the controller. The confirmation is merged with the previous speculative read in the controller and enables the controller to return data to the L1 cache sooner than a L2 cache hit. If the SCU hits, the speculative read is naturally terminated in the L2 controller, either after a certain number of cycles, or when a resource conflict exists. The L2 controller informs the SCU when a speculative read ends, either by confirmation or termination.

3.4.9 Pre-fetching Operation

The pre-fetch operation is the capability of attempting to fetch cache lines from memory in advance, to improve system performance. To enable the pre-fetch feature, the user sets bit 29 or 28 of the Auxiliary or Pre-fetch Control register. When enabled, if the slave port from the SCU receives a cacheable read transaction, a cache lookup is performed on the subsequent cache line. Bits [4:0] of the Pre-fetch Control register provide the address of the subsequent cache line. If a miss occurs, the cache line is fetched from external memory, and allocated to the L2 cache.

By default, the pre-fetch offset is 5'b00000. For example, if S0 receives a cacheable read at address 0x100, the cache line at address 0x120 is pre-fetched. Pre-fetching the next cache line might not necessarily result in optimal performance. In some systems, it might be better to pre-fetch more in

advance to achieve better performance. The pre-fetch offset enables this by setting the address of the pre-fetched cache line to Cache Line + 1 + Offset. The optimal value of the pre-fetch offset depends on the external memory read latency and on the L1 read issuing capability. The pre-fetch mechanism is not launched for a 4 KB boundary crossing.

Pre-fetch accesses can use a large number of the address slots in the controller master ports. This prevents non-prefetch accesses being serviced and affects performance. To counter this effect, the controller can drop pre-fetch accesses. This can be controlled using bit 24 of the Pre-fetch Control register. When enabled, if a resource conflict exists between pre-fetch and non-pre-fetch accesses in the controller master ports, pre-fetch accesses are dropped. When data corresponding to these dropped pre-fetch accesses returns from the external memory, it is discarded and is not allocated into the L2 cache.

3.4.10 Programming Model

The following applies to the registers used in the L2 cache controller:

- The cache controller is controlled through a set of memory-mapped registers. The memory region for these registers must be defined with strongly-ordered or device memory attributes in the L1 page tables.
- The reserved bits in all registers must be preserved; otherwise, unpredictable behavior of the device might occur.
- All registers support read and write accesses unless otherwise stated in the relevant text. A write updates the contents of a register and a read returns the contents of the register.
- All writes to registers automatically perform an initial cache sync operation before proceeding.

Initialization Sequence

As an example, a typical cache controller start-up programming sequence consists of the following register operations:

- Write to the auxiliary, tag RAM latency, data RAM latency, pre-fetch, and Power Control registers using a read-modify-write to set up global configurations:
 - Associativity and way size
 - Latencies for RAM accesses
 - Allocation policy
 - Pre-fetch and power capabilities
- Secure write to invalidate by way, offset 0x77C, to invalidate all entries in cache:
 - Write 0xFFFF to 0x77C
 - Poll cache maintenance register until invalidate operation is complete
- If required, write to register 9 to lockdown D and lockdown I.
- Write to the Interrupt Clear register to clear any residual raw interrupts set.
- Write to the interrupt mask register if it is desired to enable interrupts.
- Write to Control register 1 with the LSB set to 1 to enable the cache.

If a write is performed to the Auxiliary, Tag RAM latency, or Data RAM Latency Control register with the L2 cache enabled, a SLVERR (error) results. The L2 cache must be disabled by writing to the Control register before writing to these registers.

3.5 APU Interfaces

3.5.1 PL Co-processing Interfaces

ACP Interface

The accelerator coherence port (ACP) is a 64-bit AXI slave interface on the SCU that provides an asynchronous cache-coherent access point directly from the PL to the Cortex-A9 MP-Core processor subsystem. A range of system PL masters can use this interface to access the caches and the memory subsystem exactly the way the APU processors do to simplify software, increase overall system performance, or improve power consumption. This interface acts as a standard AXI slave and supports all standard read and write transactions without any additional coherency requirements placed on the PL components. Therefore, the ACP provides cache-coherent access from the PL to ARM caches while any memory local to the PL are non-coherent with the ARM.

Any read transactions through the ACP to a coherent region of memory interact with the SCU to check whether the required information is stored within the processor L1 caches. If it is, the data is returned directly to the requesting component. If it misses in the L1 cache, then there is also the opportunity to hit in L2 cache before finally being forwarded to the main memory. For write transactions to any coherent memory region, the SCU enforces coherence before the write is forwarded to the memory system. The transaction can also optionally allocate into the L2 cache, removing the power and performance impact of writing through to the off-chip memory.

ACP Requests

The read and write requests performed on the ACP behave differently depending on whether the request is coherent or not. This behavior is as follows:

ACP coherent read requests: An ACP read request is coherent when $ARUSER[0] = 1$ and $ARCACHE[1] = 1$ alongside $ARVALID$. In this case, the SCU enforces coherency. When the data is present in one of the Cortex-A9 processors, the data is read directly from the relevant processor, and returned to the ACP port. When the data is not present in any of the Cortex-A9 processors, the read request is issued on one of the SCU AXI master ports, along with all its AXI parameters, with the exception of the locked attribute.

ACP non-coherent read requests: An ACP read request is non-coherent when $ARUSER[0] = 1$ and $ARCACHE[1] = 1$ alongside $ARVALID$. In this case, the SCU does not enforce coherency, and the read request is directly forwarded to one of the available SCU AXI master ports to the L2 cache controller or OCM.

ACP coherent write requests: An ACP write request is coherent when $AWUSER[0] = 0$ or $AWCACHE[1] = 0$ alongside $AWVALID$. In this case, the SCU enforces coherency. When the data is

present in one of the Cortex-A9 processors, the data is first cleaned and invalidated from the relevant CPU. When the data is not present in any of the Cortex-A9 processors, or when it has been cleaned and invalidated, the write request is issued on one of the SCU AXI master ports, along with all corresponding AXI parameters with the exception of the locked attribute.

Note: The transaction can optionally allocate into the L2 cache if the write parameters are set accordingly.

ACP non-coherent write requests: An ACP write request is non-coherent when `AWUSER[0] = 0` or `AWCACHE[1] = 0` alongside `AWVALID`. In this case, the SCU does not enforce coherency, and the write request is forwarded directly to one of the available SCU AXI master ports.

ACP Usage

The ACP provides a low latency path between the PS and the accelerators implemented in the PL when compared with a legacy cache flushing and loading scheme. Steps that must take place in an example of a PL-based accelerator are as follows:

1. The CPU prepares input data for the accelerator within its local cache space.
2. The CPU sends a message to the accelerator using one of the general purpose AXI master interfaces to the PL.
3. The accelerator fetches the data via the ACP, processes the data, and returns the result via the ACP.
4. The accelerator sets a flag by writing to a known location to indicate that the data processing is complete. Status of this flag can be polled by the processor or could generate an interrupt.

Table 3-3 shows ACP read and write behavior based on current cache status. Clearly, access latency is small when cache hits occur.

When compared to a tightly-coupled coprocessor, ACP access latencies are relatively long. Therefore, ACP is not recommended for fine-grained instruction level acceleration. On the other hand, for coarse-grain acceleration such as video frame-level processing, ACP does not have a clear advantage over traditional memory-mapped PL acceleration because the transaction overhead is small relative to the transaction time, and might potentially cause undesirable cache thrashing. ACP is therefore optimal for medium-grain acceleration such as block-level crypto accelerator, video macro-block level processing, etc.

Table 3-3: ACP Read and Write Behavior

Action	Description
ACP Read - I (Invalid)	SCU fetches data from external memory through one of two AXI master interfaces. Data is forwarded to the ACP directly. It does not affect the CPU L1 cache state.
ACP Read - M (Modified)	SCU fetches data from L1 cache with M status. It does not affect the L1 cache state.
ACP Read - S (Shared)	SCU fetches data from any L1 cache with S status. It does not affect the L1 cache state.
ACP Read - E (Exclusive)	SCU fetches data from the L1 cache with E status. It does not affect the L1 cache state.

Table 3-3: ACP Read and Write Behavior (Cont'd)

Action	Description
ACP Write - I (Invalid)	Data is written to external memory through one of two AXI master interfaces. It does not affect the CPU L1 cache state.
ACP Write - M (Modified)	Data in L1 cache with M status is flushed out to external memory first. After that, ACP data is written into external memory interface. L1 cache previously with M status is changed to I status. If the SCU overwrites the entire cache line, L1 cache flush is skipped.
ACP Write - S (Shared)	Data is written to external memory through one of two AXI master interfaces. L1 cache previously with S status is changed to I state
ACP Write - E (Exclusive)	Data is written to external memory through one of two AXI master interfaces. Any L1 cache previously with S status is changed to I status.

ACP Limitations

The accelerator coherency port (ACP) has these limitations:

- Exclusive access is not allowed for coherent memory.
- Locked access is not allowed for coherent memory.
- Write transactions with length = 3, size = 3, and write strobe $\neq 11111111$ can cause the cache line in the CPUs to get corrupted.
- Continuous access to the OCM over the ACP can starve accesses from other AXI masters. To allow access from other masters, the ACP bandwidth to OCM should be moderated to less than the peak OCM bandwidth. This can be accomplished by regulating burst sizes to less than eight 64-bit words.

Note: The PS Pcore IP can be used to flag the third limitation (cache lines being corrupted). If enabled, the Xilinx ACP adapter watches for transactions that could potentially corrupt the cache and generate an error response to the master that is requesting the write request. The write transaction is allowed to proceed to the ACP interface, so the possibility of cache corruption is *NOT* eliminated. The master is notified of the possible problem in order to take the appropriate action. The ACP adapter can also generate an interrupt signal to the CPUs, which can be used by the software to detect such a situation.

Event Interface

The event bus provides a low-latency and direct mechanism to transfer status and implement a wake mechanism between the APU and the PL. The event input and output signals on this interface use toggle signaling in which an event is communicated by toggling the signal to the opposite logic level on both edges. The event bus includes these signals:

EVENTEVENTO	A toggle output signal indicating that either CPU is executing the SEV instruction.
EVENTEVENTI	A toggle input signal that wakes up either one or both CPUs if they are in a standby state initiated by the WFE instruction.

- EVENTSTANDBYWFE[1:0]** 2-level output signals indicating the state of the two CPUs. A bit is asserted if the corresponding CPU is in standby state following the execution of the WFE (wait for event) instruction.
- EVENTSTANDBYWFI[1:0]** 2-level output signals indicating the state of the two CPUs. A bit is asserted if the corresponding CPU is in standby state following the execution of the WFI (wait for interrupt) instruction.

The event bus can be used to implement PL-based accelerators. The event output can be used to trigger an ACP accelerator to read from a predefined address. Further on in the process, the event input can be used to communicate that the data has been written back over the ACP and is ready to be consumed by a CPU. A detailed description of this example follows:

1. CPU0 generates the data that is required by the accelerator in the L1/L2 cache. This data can contain both commands and information to be processed.
2. CPU0 issues an SEV (send event) instruction, causing EVENTEVENTO to toggle to the PL. The signal is connected to an accelerator IP implemented in the PL.
3. CPU0 next issues a WFE (wait For event) instruction, placing the CPU in a lower-power standby state. This is reflected in the EVENTSTANDBYWFE[0] status output to the PL.
4. The accelerator notices the toggled EVENTEVENTO signal and realizes that CPU0 is waiting. The accelerator fetches data from a prearranged address and data format via the ACP interface and begins processing.
5. After writing the result data back via the ACP, the accelerator asserts the EVENTEVENTI input to indicate that processing is complete and wakes up CPU0.
6. CPU0 wakes from its standby state, which is reflected in the EVENTSTANDBYWFE[0] output, and CPU0 continues execution using the processed data.

3.5.2 Interrupt Interface

The PS general interrupt controller (GIC) supports 64 interrupt input lines that are driven from other blocks within the PS or the PL. Six of the 64 interrupt inputs are driven from within the APU. These include the L1 parity fail, L2 interrupt (all reasons), and PMU (performance monitor unit) interrupt. The interrupt output of the GIC drives either the IRQ or FIQ inputs of each of the Cortex-A9 processors. The selection as to which processor is interrupted is accomplished via an SCU register within the APU. Table 3-4 defines the interrupts specific to the APU.

Table 3-4: APU Interrupts

Interrupt	Description
32	Any of the L1 instruction cache, L1 data cache, TLB, GHB, and BTAC parity errors from CPU 0
33	Any of the L1 instruction cache, L1 data cache, TLB, GHB, and BTAC parity errors from CPU 1
34	Any errors including parity errors from L2 controller
92	Any of the parity errors from SCU generate a third interrupt
37	Performance monitor unit (PMU) of CPU0
38	Performance monitor unit (PMU) of CPU1

3.6 Support for TrustZone Within the APU

For the system to transition from power-on reset to a stable state with both secure and non-secure applications running simultaneously, it must go through the steps depicted in [Figure 3-4](#).

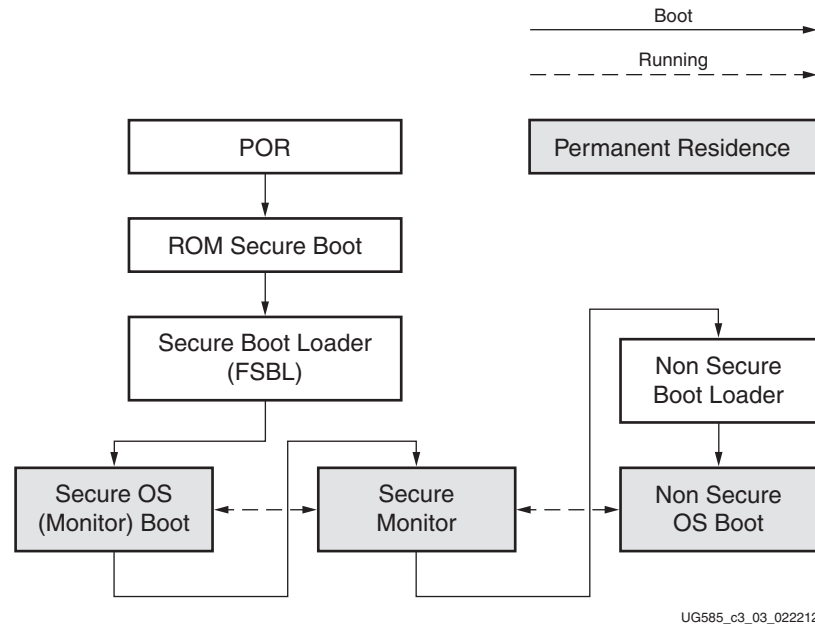


Figure 3-4: TrustZone Boot Sequence

In [Figure 3-4](#), it is assumed that the device security is on; however, this is not a necessary requirement to enable TrustZone security. In this figure, solid lines are used to show the boot flow and dotted lines are used to indicate processing transition after the system is running. The shaded blocks are software functional blocks that remain running after the system boots. In the TrustZone boot flow, Secure OS boots first and initiates a secure monitor as a secure gateway between the secure and non-secure operating systems. After the secure monitor starts, it can spawn a non-secure boot loader which in turn starts a non-secure OS. Before non-secure OS initiates, secure OS defines a set of events to force transition from non-secure OS to secure monitor. The possible events include SMC instruction, IRQ, FIQ, and data abort.

3.6.1 CPU Security Transition

Secure monitor call (SMC) causes a secure monitor exception which is available only in privileged modes. An attempt to execute this instruction in user mode causes an undefined instruction exception.

Other than entering the monitor mode through an SMC call, there are a few additional methods to allow users to switch back-and-forth between secure and non-secure worlds via the secure monitor, which acts as a gate-keeper between these two domains. These are all of the possible methods to enter the monitor mode:

- External abort handler
- FIQ handler
- IRQ handler

In the secure monitor mode, the processor is always in the secure state independent to the SCR.NS bit.

Note: Under certain circumstances, there is a possibility that Cortex-A9 TrustZone violations might trigger erroneous asynchronous aborts in secure world. For example, a secure hypervisor runs some code with SCR.NS=1 and the mask asynchronous abort bit CPSR.A=1. This non-secure code attempts to read and write memory that is marked as secure-only, and receives AXI DECERR. At that time, no exceptions occur because CPSR.A=1 and the exceptions are masked. Once secure mode is re-entered, the hypervisor switches to other code, performs DSB/ISB, and clears CPSR.A=0. In this case, the Cortex-A9 processor still remembers the pending asynchronous external abort and takes an exception as soon as CPSR.A=0.

In addition, the non-secure code might have done multi-word stores to secure memory, causing data to be modified in the L1 cache. When this data is eventually cast out, an asynchronous external abort might also be taken in the secure mode.

3.6.2 CP15 Register Access Control

There are a group of banked registers under CP15 which means that the same register has two physical copies for secure and non-secure modes. The physical register is automatically selected based on the SCR.NS bit when system is not in the secure monitor mode. In the secure monitor mode, a secure version of the physical register is always selected. Table 3-5 shows part of the CP15 registers. (Users should refer to the ARM Architecture Reference document for the entire list of the registers.)

Table 3-5: Typical CP15 Registers

CP15 Register	Banked Register	Permitted Accesses
c0	CSSELR, Cache Size Selection register	Read/write in privileged modes only
c1	SCTLR, System Control register	Read/write in privileged modes only
	ACTLR, Auxiliary Control register	Read/write in privileged modes only
c2	TTBR0, Translation Table Base 0	Read/write in privileged modes only
	TTBR0, Translation Table Base 1	Read/write in privileged modes only
	TTBCR, Translation Table Base Control	Read/write in privileged modes only
c3	DACR, Domain Access Control register	Read/write in privileged modes only
c5	DFSR, Data Fault Status register	Read/write in privileged modes only
	IFSR, Instruction Fault Status register	Read/write in privileged modes only
	ADFSR, Auxiliary Data Fault Status register	Read/write in privileged modes only
	AIFSR, Auxiliary Instruction Fault Status register	Read/write in privileged modes only
c6	DFAR, Data Fault Address register	Read/write in privileged modes only
	IFAR, Instruction Fault Address register	Read/write in privileged modes only
c7	PAR, Physical Address register (VA to PA translation)	Read/write in privileged modes only

Table 3-5: Typical CP15 Registers (Cont'd)

CP15 Register	Banked Register	Permitted Accesses
c10	PRRR, Primary Region Remap register	Read/write in privileged modes only
	NMRR, Normal Memory Remap register	Read/write in privileged modes only
c12	VBAR Vector Base Address register	Read/write in privileged modes only

Table 3-6 shows additional registers with access control at secure and non-secure states.

Table 3-6: CP15 State Control Registers

CP15 Register	Secure Register	Permitted Accesses
c1	NSACR, Non-Secure Access Control register	Read/write in privileged modes Read-only in non-secure privileged modes
	SCR, Secure Configuration register	Read/write in privileged modes
	SDER, Secure Debug Enable register	Read/write in privileged modes
c12	MVBAR, Monitor Vector Base Address register	Read/write in privileged modes

3.6.3 MMU Security

The MMU in Cortex A9 is enhanced with TrustZone features to provide access permission checks in addition to address translation. In each of the secure and non-secure worlds, a single set of two-level page tables stored in main memory controls the contents of the instruction- and data-side translation look-aside buffers (TLBs). The calculated physical address associated with the virtual address is placed in the TLB, accompanied by an NSTID (non-secure table identifier) that allows secure and non-secure entries to co-exist. The TLBs are enabled in each world through a single bit in CP15 Control register c1, providing a single address translation and protection scheme for software.

This is how the TLB and BTAC state is handled when transitioning between secure and non-secure worlds:

- BPIALL, when executed from the secure state, might or might not invalidate BTAC entries in the non-secure state.
- TLBIALL, when executed from the secure state, does not invalidate BTAC entries from the non-secure state.
- Writing to CONTEXTIDR from the secure state, but with SCR.NS=1, does invalidate BTC entries in the non-secure state.

Because any context switch writes to CONTEXTIDR and causes the non-secure BTAC entries to be invalidated, BPIALL from secure state not affecting non-secure BTAC entries is not going to be a problem.

3.6.4 L-1 Cache Security

Each cache line can contain secure or non-secure data. The effect of an access attempting to violate data security causes a cache miss. On a miss, the next step is to go to the external memory which returns an abort if the NS attribute does not match access permissions.

3.6.5 Security Exception Control

When an exception is taken, processor execution is forced to an address that corresponds to the type of exception. These addresses are called the exception vectors. By default, the exception vectors are eight consecutive word-aligned memory addresses, starting at an exception base address as follows:

There are three exception base addresses for Zynq-7000 EPP devices:

- The non-secure exception base address is used for all exceptions processed in non-secure state.
- The secure exception base address is used for all exceptions processed in secure state, but not in monitor mode.
- The monitor exception base address is used for all exceptions processed in monitor mode.

3.6.6 CPU Debug TrustZone Access Control

Four control signals control CPU debug status: DBGEN, NIDEN, SPIDEN, and SPNIDEN. These four control signals are part of a secure and protected register within the device configuration interface module. The most important modes supported are identified in [Table 3-7](#).

Table 3-7: CPU Debug TrustZone Access Control

Mode	DBGEN	NIDEN	SPIDEN	SPNIDEN	Comment
No Debug	0	0	0	0	No CPU debug at all
Non-Secure Non Invasive Debug	1	1	0	0	Allow non-invasive debug, such as Trace and Performance Monitor, in Non-Secure mode
Non-Secure Invasive Debug	1	1	0	0	Allow invasive debug, such as stop processor, in Non-Secure mode
Secure Non Invasive Debug	1	1	0	1	This allow CPU Trace and Profile under secure condition
Secure Invasive Debug	1	1	1	1	This allows Invasive debug for secure mode

3.6.7 SCU Register Access Control

The SCU non-secure Access Control register (SNACR) controls the global non-secure accesses for each major component within the SCU. The Interrupt Controller Distributor Control register (ICDDCR) is a banked register for controlling secure and non-secure accesses.

3.6.8 TrustZone Support in the L2 Cache

The cache controller attaches an NS bit to all data stored in the L2 cache and in internal buffers. A non-secure transaction cannot access secure data. Therefore the controller treats secure and non-secure data as being part of two different memory spaces. The controller treats a non-secure access to secure data in the L2 cache as a miss. For a read transfer, the cache controller sends a line fill command to external memory, propagates any security errors from external memory to the processor, and does not allocate the line in L2.

These are a few notes about the Trustzone support in L2:

- The L2 Control register can only be written to with an access tagged as secure, to enable or disable the L2 cache.
- The Auxiliary Control register can only be written to with an access tagged as secure. Bit [26] in the Auxiliary Control register is for NS lockdown enable. This bit should be used to determine whether non-secure accesses can modify a lockdown register.
- Non-Secure maintenance operations do not clean or invalidate secure data.

3.7 Application Processing Unit (APU) Reset

3.7.1 Reset Functionality

The APU supports different reset modes that enable the user to reset different parts of the block independently. Applicable resets and their functions are as follows:

Power-on Reset	The power-on reset or cold reset is applied when the power is first applied to the system or through the PS_POR_B device pin. In this reset mode, both CPUs, the NEON coprocessors, and the debug logic is reset.
System Reset	A system reset initializes the Cortex-A9 processor and the NEON coprocessors, apart from the debug logic. Break points and watch points are retained during this reset. This reset is applied through the PS_SRST_B device pin.
Software Reset	A software or warm reset initializes the Cortex-A9 processor and the NEON coprocessors, apart from the debug logic. Break points and watch points are retained during this reset. Processor reset is typically used for resetting a system that has been operating for some time. This reset is applied through the A9_CPU_RST_CTRL.A9_RSTx register.
System Debug Reset	This reset is similar to the software reset; however, it is triggered through the JTAG interface.
Debug Reset	This reset initializes the debug logic in a Cortex-A9 processor, including break point and watch point values. It is triggered through the JTAG interface.

Note: The APU in Zynq-7000 EPP devices does not support an independent reset for the NEON coprocessors.

Note: Unlike the POR or system resets, when the user applies a software reset to a single processor, the user must stop the associated clock, de-assert the reset, and then restart the clock. During a system or POR reset, hardware automatically takes care of this. Assuming the user wants to reset CPU0, the user must set the following fields in the SLCR.A9_CPU_RST_CTRL (address 0x000244) register in the order listed:

1. A9_RST0 = 1 to assert reset to CPU0
2. A9_CLKSTOP0 = 1 to stop clock to CPU0

3. A9_RST0 = 0 to release reset to CPU0
4. A9_CLKSTOP0 = 0 to restart clock to CPU0

3.7.2 APU State After Reset

Table 3-8 summarizes the state of the APU after the reset.

Table 3-8: APU State after Reset

Function	State after Reset
CPU1	Kept in a WFE state while executing code located at address 0xFFFFFE00 to 0xFFFFFFFF0
L1 Caches	Disabled
Validation	Unknown (requires invalidation prior to usage)
MMUs	Disabled
SCU	
Address Filtering	Upper and lower 1M addresses within the 4G address space are mapped to OCM and the rest of the addresses are routed to the L2
L2 Cache	Disabled
L2 wait-states	Tag RAM and Data RAM wait states are both 7-7-7 for setup latency, write access latency, and read access latency
Validation	Unknown (requires invalidation prior to usage)

3.8 Power Considerations

3.8.1 Introduction

The APU incorporates many features to improve its dynamic power efficiency:

- Either of the CPUs can be put in the sleep mode and waken up on events and interrupts.
- The L2 cache can be put in the standby mode when the CPUs are in that mode.
- Clock gating is extensively used in all the sub-blocks within the module.
- Accurate branch and return prediction reduces the number of incorrect instruction fetch and decode operations.
- Physically-addressed caches reduce the number of cache flushes and refills, saving energy in the system.
- The CPUs implement micro TLBs for local address translation which reduces the power consumed in translation and protection look-ups.
- The tag RAMs and data RAMs are accessed sequentially to eliminate accesses to the unwanted data RAMs and thus minimize unnecessary power consumption.
- To reduce power consumption in the L1 caches, the number of full cache reads is reduced by taking advantage of the sequential nature of memory accesses. If a cache read is sequential to

the previous cache read, and the address is within the same cache line, only the data RAM set that was previously read is accessed.

- If an instruction loop fits in four BTAC entries, then instruction cache accesses are turned off in order to lower power consumption.
- The clock to the NEON Engine is dynamically controlled by the CPU and the engine gets clocked only when a NEON instruction is issued.

Note: Power to the APU or any of its sub-blocks cannot be turned off while the PS is powered on.

3.8.2 Standby Mode

In the standby mode of operation, the device is still powered-up, but most of its clocks are gated off. This means that the processor is in a static state and the only power drawn is due to leakage currents and the clocking of the small amount of logic which looks out for the wakeup condition.

This mode is entered using either the WFI (wait for interrupt) or WFE (wait for event) instructions. It is recommended that a DSB memory barrier be used before WFI or WFE, to ensure that pending memory transactions complete.

The processor stops execution until a wake up event is detected. The wake up condition is dependent on the entry instruction. For WFI, an interrupt or external debug request wakes the processor. For WFE, a number of specified events exist, including another processor in an MP system executing the SEV instruction. A request from the SCU can also wake up the clock for a cache coherency operation in an MP system. This means that the cache of a processor which is in standby state continues to be coherent with caches of other processors. A processor reset always forces the processor to exit from the standby condition.

The standby mode of the L2 cache controller can be enabled by setting bit 0 of the L2 controller Power Control register. This mode is used in conjunction with the wait state (WFI/WFE) of the processor that drives the controller. Before entering the wait state, the Cortex-A9 processor must set its status field in the CPU Power Status register of the SCU to signal its entering sleep mode. The Cortex-A9 processor then executes a WFI or WFE entry instruction. The SCU CPU Power Status register bits can also be read by a Cortex-A9 processor exiting low-power mode to determine its state before executing its reset setup.

If the MP system is in the standby mode, the SCU signals to the L2 cache controller to gate its clock and the controller honors that when the L2 becomes idle. Any transaction from the SCU to the L2 restarts the clock and triggers a response with 2-3 clock cycles of delay.

3.8.3 Dynamic Clock Gating in the L2 Controller

Bit 1 of the L2 Controller Power Control register enables the dynamic clock gating feature within the controller. If this feature is enabled, the cache controller stops its clock when it is idle for 32 clock cycles. The controller stops the clock until there is a transaction on its slave interface from the SCU. If this interface detects a transaction, it restarts its clock and accepts the new transaction with 2-3 cycles of delay.

System Addresses

4.1 Address Map

The comprehensive system level address map is shown in [Table 4-1](#). The shaded entries indicate that the address range is reserved and should not be accessed. [Table 4-2](#) identifies reserved address ranges.

Table 4-1: System-Level Address Map

Address Range	CPU's and ACP	AXI_HP	Other Bus Masters ⁽¹⁾	Notes
0000_0000 to 0003_FFFF ⁽²⁾	OCM	OCM	OCM	Address not filtered by SCU and OCM is mapped low
	DDR	OCM	OCM	Address filtered by SCU and OCM is mapped low
	DDR			Address filtered by SCU and OCM is not mapped low
				Address not filtered by SCU and OCM is not mapped low
0004_0000 to 0007_FFFF	DDR			Address filtered by SCU
				Address not filtered by SCU
0008_0000 to 000F_FFFF	DDR	DDR	DDR	Address filtered by SCU
		DDR	DDR	Address not filtered by SCU ⁽³⁾
0010_0000 to 3FFF_FFFF	DDR	DDR	DDR	Accessible to all interconnect masters
4000_0000 to 7FFF_FFFF	PL		PL	General Purpose Port #0 to the PL, M_AXI_GP0
8000_0000 to BFFF_FFFF	PL		PL	General Purpose Port #1 to the PL, M_AXI_GP1
E000_0000 to E02F_FFFF	IOP		IOP	I/O Peripheral registers, see Table 4-6
E100_0000 to E5FF_FFFF	SMC		SMC	SMC Memories, see Table 4-6
F800_0000 to F800_0BFF	SLCR		SLCR	SLCR registers, see Table 4-6
F800_1000 to F880_FFFF	PS		PS	PS System registers, see Table 4-6
F890_0000 to F8F0_2FFF	CPU			CPU Private registers, see Table 4-6
FC00_0000 to FDFE_FFFF ⁽⁴⁾	Quad-SPI		Quad-SPI	Quad-SPI linear address for linear mode
FFFC_0000 to FFFF_FFFF ⁽²⁾	OCM	OCM	OCM	OCM is mapped high
				OCM is not mapped high

Notes:

1. The other bus masters include the S_AXI_GP interfaces, Device configuration interface (DevC), DAP controller, DMA controller and the various controllers with local DMA units (Ethernet, USB and SDIO).
2. The OCM is divided into four 64 KB sections. Each section is mapped independently to either the low or high addresses ranges, but not both at the same time. In addition, the SCU can filter addresses destined for the OCM low address range to the DDR DRAM controller instead. A detailed discussion of the OCM is explained in [Chapter 29, On-Chip Memory \(OCM\)](#).
3. For each 64 KB section mapped to the high OCM address range via slcr.OCM_CFG[RAM_HI] which is not also part of the SCU address filtering range will be aliased for CPU and ACP masters at a range of (0x000C_0000 to 0x000F_FFFF). See [Chapter 29, On-Chip Memory \(OCM\)](#) for more information.
4. When a single device is used, it must be connected to QSPI 0. In this case, the address map starts at FC00_0000 and goes to a maximum of FCFE_FFFF (16 MBs). When two devices are used, both devices must be the same capacity. The address map for two devices depends on the size of the devices and their connection configuration. For the shared 4-bit parallel I/O bus, the QSPI 0 device starts at FC00_0000 and goes to a maximum of FCFE_FFFF (16 MBs). The QSPI 1 device starts at FD00_0000 and goes to a maximum of FDFE_FFFF (another 16 MBs). If the first device is less than 16 MBs in size, then there will be a memory space hole between the two devices. For the 8-bit dual stacked mode (8-bit bus), the memory map is continuous from FC00_0000 to a maximum of FDFE_FFFF (32 MBs).

Table 4-2: System-Level Address Map (Reserved Addresses)

Address Range	CPUs and ACP	AXI_HP	Other Bus Masters ⁽¹⁾	Notes
C000_0000 to DFFF_FFFF				Reserved
E030_0000 to E0FF_FFFF				Reserved
E600_0000 to F7FF_FFFF				Reserved
F800_0C00 to F800_0FFF				Reserved
F801_0000 to F88F_FFFF				Reserved
F8F0_3000 to FBFF_FFFF				Reserved
FE00_0000 to FFFB_FFFF				Reserved

PL AXI Interface Note

There are two general purpose interconnect ports that go to the PL, M_AXI_GP{1,0}. Each port is addressable by masters in the PS and each port occupies 1 GB of system address space in the ranges specified in [Table 4-1](#). The M_AXI_GP addresses are directly from the PS; they are not remapped on their way to the PL. The addresses outside of these ranges are not presented to the PL.

Execute-In-Place Capable Devices

The following devices are execute-in-place capable:

- DDR
- OCM
- SMC SRAM/NOR
- Quad-SPI (linear addressing mode)
- M_AXI_GP{1, 0} (PL block RAM with a suitable PL controller)

4.2 System Bus Masters

The CPUs and AXI_ACP see the same memory map, except the CPUs have a private bus to access their private timer, interrupt controller, and shared L2 cache / SCU registers. The AXI_HP interfaces provide high bandwidth to the DDR DRAM and OCM. The other system bus masters include:

- DMA controller, see [Chapter 9, DMA Controller](#)
- Device configuration interface (DevC), see [Chapter 6, Boot and Configuration](#)
- Debug access port (DAP), see [Chapter 28, System Test and Debug](#)
- PL bus master controllers attached to AXI general purpose ports (S_AXI_GP[1:0]), see [Chapter 5, Interconnect](#) and [Chapter 21, Programmable Logic Description](#)
- AHB bus master ports with local DMA units (Ethernet, USB, and SDIO)

4.3 SLCR Registers

The System-Level Control registers (SLCR) consist of various registers that are used to control the PS behavior. These registers are accessible via the central interconnect using load and store instructions. The detailed descriptions for each register can be found in [Appendix B, Register Details](#). A summary of the SLCR registers with their base addresses is shown in [Table 4-3](#).

Table 4-3: SLCR Register Map

Register Base Address	Description	Reference
F800_0000	SLCR write protection lock and security	
F800_0100	Clock control and status	See Chapter 25, Clocks
F800_0200	Reset control and status	See Chapter 26, Reset System
F800_0300	APU control	See Chapter 3, Application Processing Unit
F800_0400	TrustZone control	
F800_0500	CoreSight SoC debug control	See Chapter 28, System Test and Debug
F800_0600	DDR DRAM controller	See Chapter 10, DDR Memory Controller

Table 4-3: SLCR Register Map (Cont'd)

Register Base Address	Description	Reference
F800_0700	MIO pin configuration	See Chapter 2, Signals, Interfaces, and Pins
F800_0800	MIO parallel access	See Chapter 2, Signals, Interfaces, and Pins
F800_0900	Miscellaneous control	See Chapter 29, On-Chip Memory (OCM)
F800_0A00	On-chip memory (OCM) control	See Chapter 29, On-Chip Memory (OCM)
F800_0B00	I/O buffers for MIO pins (GPIOB) and DDR pins (DDRIOB)	See Chapter 2, Signals, Interfaces, and Pins

4.4 CPU Private Bus Registers

The registers shown in [Table 4-4](#) are only accessible by the CPU on the CPU private bus. The accelerator coherency port (ACP) cannot access any of the private CPU registers. The private CPU registers are used to control subsystems in the APU.

Table 4-4: CPU Private Register Map

Register Base Address	Description
F890_0000 to F89F_FFFF	Top-level interconnect configuration and Global Programmers View (GPV)
F8F0_0000 to F8F0_00FC	SCU control and status
F8F0_0100 to F8F0_01FF	Interrupt controller CPU
F8F0_0200 to F8F0_02FF	Global timer
F8F0_0600 to F8F0_06FF	Private timers and private watchdog timers
F8F0_1000 to F8F0_1FFF	Interrupt controller distributor
F8F0_2000 to F8F0_2FFF	L2-cache controller

4.5 SMC Memory

The SMC memories are accessed via a 32-bit AHB bus (see [Table 4-5](#)). The SMC control registers are listed [Table 4-6](#). Refer to [Chapter 11, Static Memory Controller](#) for information on the functionality of the NAND and SRAM/NOR controllers.

Table 4-5: SMC Memory Address Map

Register Base Address	Description
E100_0000	SMC NAND Memory address range
E200_0000	SMC SRAM/NOR CS 0 Memory address range
E400_0000	SMC SRAM/NOR CS 1 Memory address range

4.6 PS I/O Peripherals

The I/O Peripheral registers are accessed via a 32-bit APB bus, shown in [Table 4-6](#).

Table 4-6: I/O Peripheral Register Map

Register Base Address	Description
E000_0000, E000_1000	UART Controllers 0, 1
E000_2000, E000_3000	USB Controllers 0, 1
E000_4000, E000_5000	I2C Controllers 0, 1
E000_6000, E000_7000	SPI Controllers 0, 1
E000_8000, E000_9000	CAN Controllers 0, 1
E000_A000	GPIO Controller
E000_B000, E000_C000	Ethernet Controllers 0, 1
E000_D000	Quad-SPI Controller
E000_E000	Static Memory Controller (SMC)
E010_0000, E010_1000	SDIO Controllers 0, 1

4.7 Miscellaneous PS Registers

The PS system registers are accessed via a 32-bit AHB bus (see [Table 4-7](#)).

Table 4-7: PS System Register Map

Register Base Address	Description
F800_1000, F800_2000	Triple timer counter 0, 1
F800_3000	DMAC when secure
F800_4000	DMAC when non-secure
F800_5000	System watchdog timer (SWDT)
F800_6000	DDR DRAM controller
F800_7000	Device configuration interface (DevC)
F800_8000	AXI_HP 0 high performance AXI interface w/ FIFO
F800_9000	AXI_HP 1 high performance AXI interface w/ FIFO
F800_A000	AXI_HP 2 high performance AXI interface w/ FIFO
F800_B000	AXI_HP 3 high performance AXI interface w/ FIFO
F800_C000	On-chip memory (OCM)
F800_D000	Reserved
F880_0000	CoreSight debug control

Interconnect

5.1 Introduction

The interconnect, located within the PS, comprises multiple switches to connect system resources using AXI point-to-point channels for communicating addresses, data and response transactions between master and slave clients. This ARM AMBA 3.0 interconnect implements a full array of the interconnect communications capabilities and overlays for QoS, debug and test monitoring. The interconnect manages multiple outstanding transactions and is architected for low-latency paths for the ARM CPUs and, for the PL master controllers, a high-throughput and cache coherent data paths.

5.1.1 Features

The interconnect is the primary mechanism for data communications. The following summarizes the interconnect features:

- The interconnect is based on AXI high performance datapath switches:
 - Snoop control unit
 - L2 cache controller
 - Interconnect switches based on ARM NIC-301
 - Central interconnect
 - Master interconnect
 - Slave interconnect
 - Memory interconnect
 - OCM interconnect
 - AHB and APB bridges
- PS-PL Interfaces
 - AXI_ACP, one cache coherent master port for the PL
 - AXI_HP, four high performance/bandwidth master ports for the PL
 - AXI_GP, four general purpose ports (two master ports and two slave ports)

5.1.2 Block Diagram

Figure 5-1 shows the block diagram for the interconnect.

Interconnect Masters

The interconnect masters are shown at the top of [Figure 5-1](#).

The interconnect masters include:

- CPUs and advanced coherency port (ACP)
- High performance PL Interfaces, AXI_HP{3:0}
- General purpose PL interfaces, AXI_GP{1:0}
- DMA controller
- AHB masters (I/O peripherals with local DMA units)
- Device configuration (DevC) and debug access port (DAP)

Snoop Control Unit (SCU)

The functionality of the snoop control unit is described in [Chapter 3, Application Processing Unit](#). The address filtering feature of the SCU makes the SCU function like a switch from the perspective of the traffic from its AXI slave ports to its AXI master ports.

Central Interconnect

The central interconnect is the core of the ARM NIC301-based interconnect switches.

Master Interconnect

The master interconnect switches the low-to-medium speed traffic from AXI_GP ports, DevC and DAP to the central interconnect.

Slave Interconnect

The slave interconnect switches the low-to-medium speed traffic from the central interconnect to AXI_GP, I/O peripherals (IOP) and other blocks.

Memory Interconnect

The memory interconnect switches the high speed traffic from the AXI_HP ports to DDR DRAM and on-chip RAM (via another interconnect).

OCM Interconnect

The OCM interconnect switches the high speed traffic from the central interconnect and the memory interconnect.

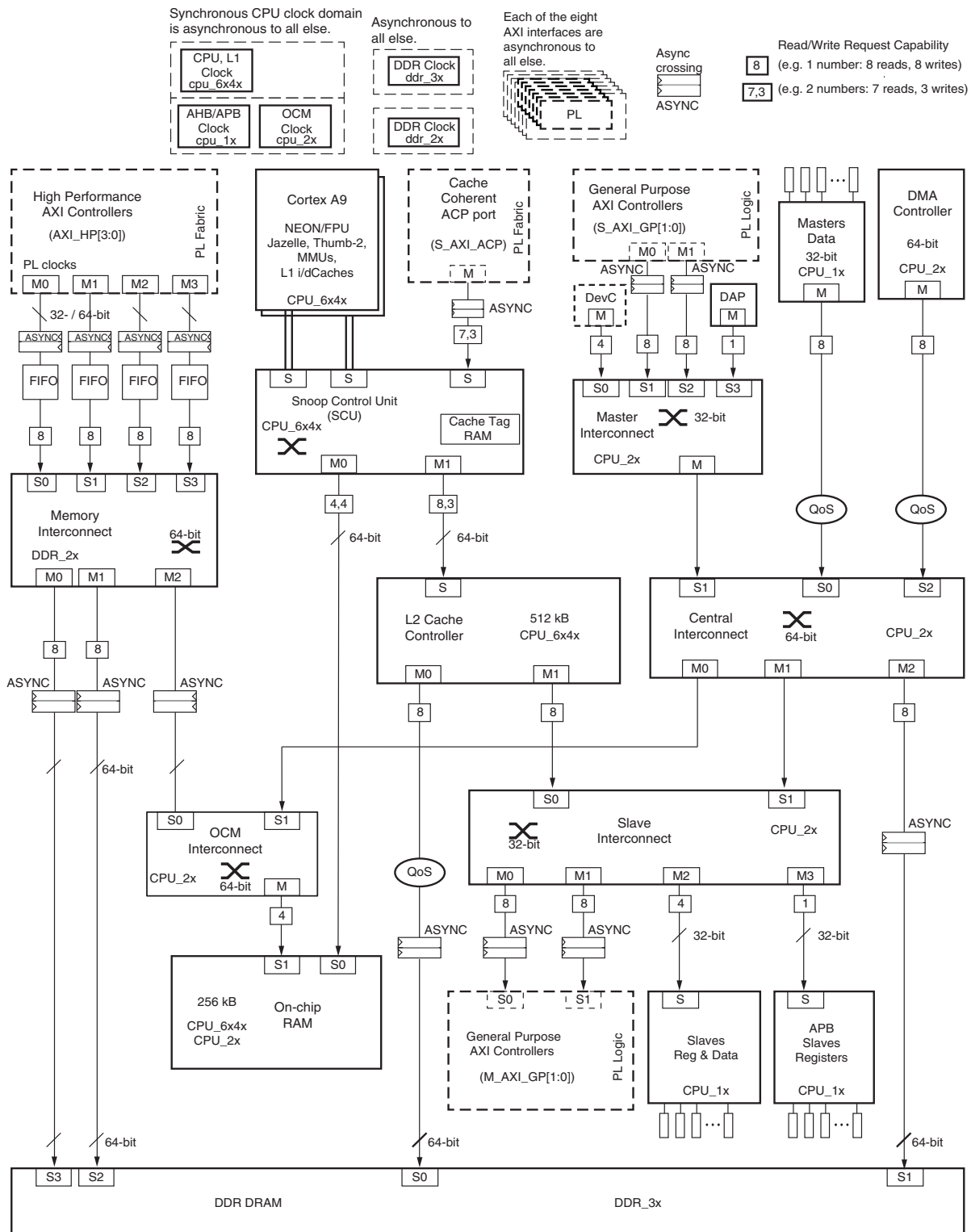


Figure 5-1: Interconnect Block Diagram

L2 Cache Controller

The functionality of the L2 cache controller is described in [Chapter 3, Application Processing Unit](#). The address filtering feature of the L2 cache controller makes the L2 cache controller function like a switch from the perspective of the traffic from its AXI slave ports to its AXI master ports.

Interconnect Slaves

The interconnect slaves are shown toward the bottom of [Figure 5-1](#). The Interconnect slaves include:

- On-chip RAM (OCM)
- DDR DRAM
- General purpose PL interfaces, M_AXI_GP{1:0}
- AHB slaves (IOP with local DMA units)
- APB slaves (registers in blocks inside the IOP)
- GPV (programmable registers of the interconnect, not shown in the figure)

5.1.3 Datapaths

[Table 5-1](#) lists the major datapaths used by the PS interconnect.

Table 5-1: Interconnect Datapaths

Source	Destination	Type	Clock at source	Clock at destination	Sync or Async ⁽¹⁾	Data width	R/W Request Capability	Advanced QoS
CPU	SCU	AXI	CPU_6x4x	CPU_6x4x	Sync	64	7, 12	-
AXI_ACP	SCU	AXI	SAXIACPCLK	CPU_6x4x	Async	64	7, 3	-
AXI_HP	FIFO	AXI	SAXIHPnACLK	DDR_2x	Async	32/64	14-70, 8-32 ⁽²⁾	-
S_AXI_GP	Master IC.	AXI	SAXIGPnACLK	CPU_2x	Async	32	8, 8	-
DevC	Master IC.	AXI	CPU_1x	CPU_2x	Sync	32	8, 4	-
DAP	Master IC.	AHB	CPU_1x	CPU_2x	Sync	32	1, 1	-
AHB masters	Central IC.	AXI	CPU_1x	CPU_2x	Sync	32	8, 8	X
DMA Controller	Central IC.	AXI	CPU_2x	CPU_2x	Sync	64	8, 8	X
Master IC.	Central IC.	AXI	CPU_2x	CPU_2x	Sync	64	-	-
FIFO	Memory IC.	AXI	DDR_2x	DDR_2x	Sync	64	8, 8	-
SCU	L2 Cache	AXI	CPU_6x4x	CPU_6x4x	Sync	64	8, 3	-
Memory IC.	OCM IC.	AXI	DDR_2x	CPU_2x	Async	64	-	-
Central IC.	OCM IC.	AXI	CPU_2x	CPU_2x	Sync	64	-	-
L2 Cache	Slave IC.	AXI	CPU_6x4x	CPU_2x	Sync	64	8, 8	-
Central IC.	Slave IC.	AXI	CPU_2x	CPU_2x	Sync	64	-	-

Table 5-1: Interconnect Datapaths (Cont'd)

Source	Destination	Type	Clock at source	Clock at destination	Sync or Async ⁽¹⁾	Data width	R/W Request Capability	Advanced QoS
SCU	On-chip RAM	AXI	CPU_6x4x	CPU_2x	Sync	64	4, 4	-
OCM IC.	On-chip RAM	AXI	CPU_2x	CPU_2x	Sync	64	4, 4	-
Slave IC.	APB slaves	APB	CPU_2x	CPU_1x	Sync	32	1, 1	-
Slave IC.	AHB slaves	AXI	CPU_2x	CPU_1x	Sync	32	4, 4	-
Slave IC.	AXI_GP	AXI	CPU_2x	MAXIGPnACLK	Async	32	8, 8	-
L2 Cache	DDR Controller	AXI	CPU_6x4x	DDR_3x	Async	64	8, 8	X
Central IC.	DDR Controller	AXI	CPU_2x	DDR_3x	Async	64	8, 8	-
Memory IC.	DDR Controller	AXI	DDR_2x	DDR_3x	Async	64	8, 8	-
Slave IC.	GPV	-(3)	CPU_2x	(multiple)	-	-	-	-

Notes:

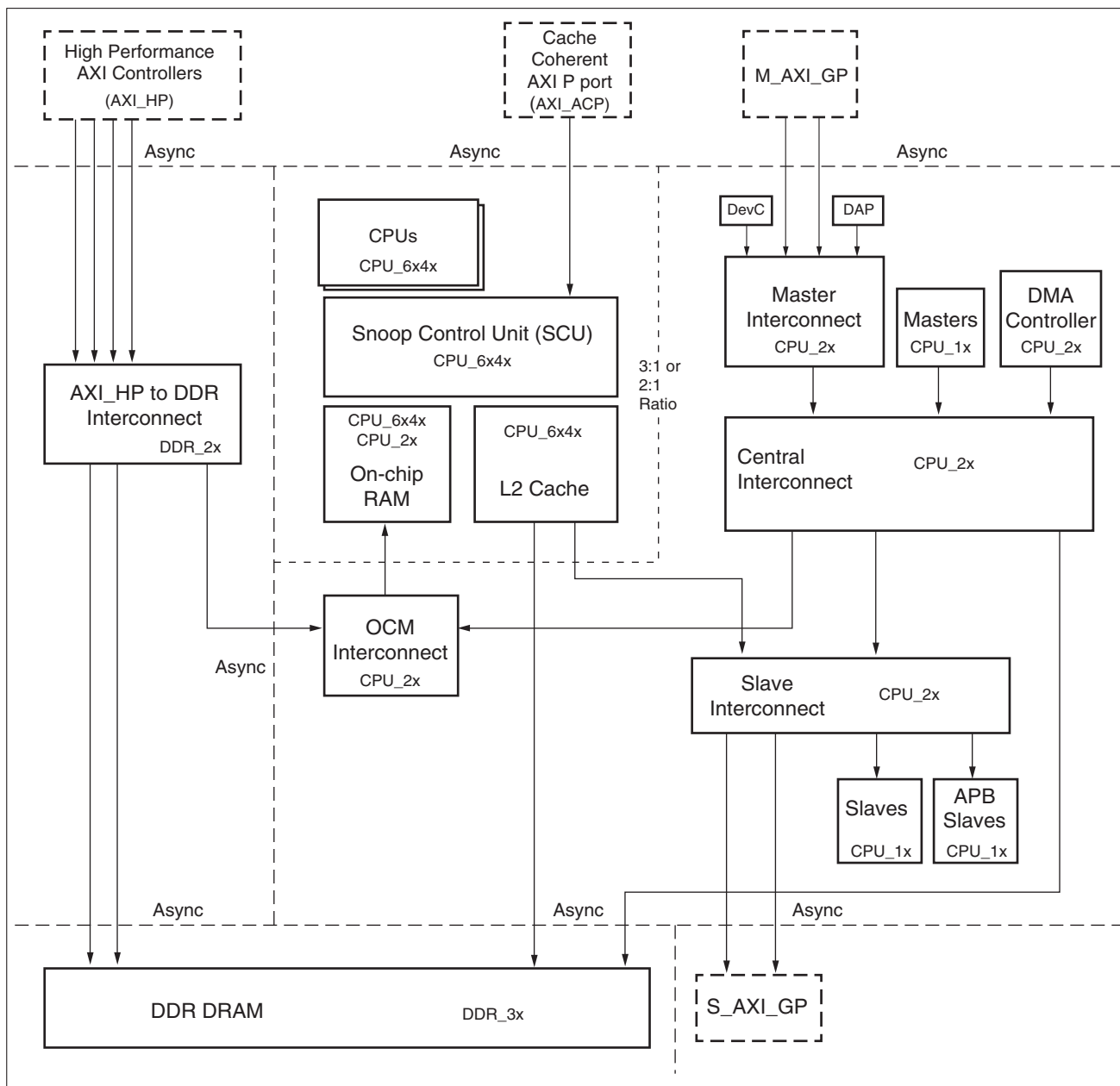
- Each asynchronous path includes an asynchronous bridge for clock domain crossing.
- Burst-length dependent (see [AXI_HP Interfaces](#)).
- The path from the slave interconnect to GPV is an internal path within the entire interconnect structure. When accessing GPV, ensure that all clocks are on (see [Clocks and Resets](#)).

5.1.4 Clock Domains

The interconnect, masters, and slaves use these clocks:

- CPU_6x4x: CPUs, SCU, L2 cache controller, on-chip RAM
- CPU_2x: Central interconnect, master interconnect, slave interconnect, OCM interconnect
- CPU_1x: AHB masters, AHB slaves, APB slaves, DevC, DAP
- DDR_3x: DDR Memory Controller
- DDR_2x: Memory interconnect, FIFOs
- SAXIACPACLK: AXI_ACP slave port
- SAXIHP0ACLK: AXI_HP0 slave port
- SAXIHP1ACLK: AXI_HP1 slave port
- SAXIHP2ACLK: AXI_HP2 slave port
- SAXIHP3ACLK: AXI_HP3 slave port
- SAXIGP0ACLK: AXI_GP0 slave port
- SAXIGP1ACLK: AXI_GP1 slave port
- MAXIGP0ACLK: AXI_GP0 master port
- MAXIGP1ACLK: AXI_GP1 master port

Except for CPU_6X4X, CPU_2X, and CPU_1X, which are synchronous clocks with a ratio of 6:2:1 or 4:2:1, all clocks in the above list are asynchronous to one another, as shown in Figure 5-2.



UG585_c5_02_050212

Figure 5-2: Interconnect Clock Domains

5.1.5 Connectivity

The interconnect is not a full cross-bar structure. Table 5-2 shows which master can access which slave.

Table 5-2: Master - Slave Access

Master	Slave	On-chip RAM	DDR Port 0	DDR Port 1	DDR Port 2	DDR Port 3	M_AXI_GP	AHB Slaves	APB Slaves	GPV
CPU		X	X				X	X	X	X
AXI_ACP		X	X				X	X	X	X
AXI_HP{0,1}		X				X				
AXI_HP{2,3}		X			X					
S_AXI_GP{0,1}		X		X			X	X	X	
DMA Controller		X		X			X	X	X	
AHB Masters		X		X			X	X	X	
DevC, DAP		X		X			X	X	X	

5.1.6 AXI ID

The interconnect uses 13-bit AXI IDs, consisting of (from MSB to LSB):

- 3 bits that identify the interconnect (central, master, slave, etc.)
- 8 bits supplied by the master; width is determined by the largest AXI ID width from all masters
- 2 bits that identify the slave interface of the identified interconnect

Table 5-3 lists all possible AXI ID values that a slave can observe.

Table 5-3: Slave Visible AXI ID Values

Master	Master ID width	AXI ID (as seen by the slaves)
AXI_HP0	6	13'b00000xxxxxx00
AXI_HP1	6	13'b00000xxxxxx01
AXI_HP2	6	13'b00000xxxxxx10
AXI_HP3	6	13'b00000xxxxxx11
DMAC controller	4	13'b0010000xxxx00
AHB masters	3	13'b00100000xxx01
DevC	0	13'b0100000000000
DAP	0	13'b0100000000001

Table 5-3: Slave Visible AXI ID Values (Cont'd)

Master	Master ID width	AXI ID (as seen by the slaves)
S_AXI_GP0	6	13'b01000xxxxxx10
S_AXI_GP1	6	13'b01000xxxxxx11
CPUs, AXI_ACP via L2 M1 port	8	13'b011xxxxxxxx00
CPUs, AXI_ACP via L2 M0 port	8	13'b100xxxxxxxx00
Notes: x, which can be either 0 or 1, originates from the requesting master.		

5.1.7 Read/Write Request Capability

The R/W Request Capability shown in Figure 5-1 and in Table 5-1 describes the maximum number of requests that the master of a datapath can issue. This does not mean the master can always issue that maximum number of requests under any circumstances or scenarios. There are conditions where other limiting factors can be active to reduce the number of requests.

One particular example is the extended write rule in the deadlock avoidance scheme, which ensures the network only issues a write transaction (on the AW channel) if all the outstanding write transactions have had the last write data beat transmitted (on the W channel). Under this rule, if the number of write data beats is large, preventing a second write request from being issued in a certain spot in the network, because the network must wait until the last beat of write data of the first write is transmitted. In this case, only a single write request can be issued by a master.

5.1.8 Register Overview

Table 5-4 provides an overview of the GPV registers.

Table 5-4: GPV Register Overview

Function	Name	Overview
TrustZone	security_fssw_s0 security_fssw_s1	Control boot secure settings for the slave ports of the slave interconnect.
Advanced QoS	qos_cntl, max_ot, max_comb_ot, aw_p, aw_b, aw_r, ar_p, ar_b, ar_r	Control advanced QoS features, maximum number of outstanding transactions, AW and AR channel peak rates, burstiness, average rates.

5.2 Quality of Service

5.2.1 Basic Arbitration

Each interconnect (central, master, slave, memory) uses a two-level arbitration scheme to resolve contention. The first-level arbitration is based on the priority indicated by the AXI QoS signals from the master or programmable registers. The highest QoS value has the highest priority. The second-level arbitration is based on a least recently granted (LRG) scheme and is used when multiple requests are pending with the same QoS signal value. Information on OCM arbitration can be found in [Chapter 10, DDR Memory Controller](#).

5.2.2 Advanced QoS

In addition to the basic arbitration, the interconnect provides an advanced QoS control mechanism. This programmable mechanism influences interconnect arbitration for requests from these masters:

- CPUs and ACP requests to DDR (via L2 cache controller port M0)
- DMA controller requests to DDR and OCM (via the central interconnect)
- AMB master requests to DDR and OCM (via the central interconnect)

In the PS, advanced QoS modules exist on the following paths:

- Path from L2 cache to DDR
- Path from DMA controller to the central interconnect
- Path from AHB masters to the central interconnect

The QoS module is based on ARM's QoS-301, which is an extension to the NIC-301 network interconnect. They provide facilities to regulate transactions as follows:

- Maximum number of outstanding transactions
- Peak rates,
- Average rates
- Burstiness

For more information, refer to *CoreLink QoS-301 Network Interconnect Advanced Quality of Service Technical Reference Manual*.

The usage of QoS arbitration for all slave interfaces should be used with careful deliberation, as fixed priority arbitration leads to starvation issues if not used properly. By default, all ports have equal priority so starvation is not an issue.

Rationale

The user is expected to create "well behaved" masters in the PL, which sufficiently throttle their rate of command issuance, or use the AXI_HP issuance capability settings. However, traffic from CPUs

(through L2 cache), the DMA controller, and the IOP masters can interfere with traffic from the PL. The QoS modules allow users to throttle these PS masters to ensure expected/consistent throughput and latency for the user design in the PL or specific PS masters. This is especially useful for video, which requires guaranteed maximum latency. By regulating the “irregular” masters like CPUs, the DMA controller, and IOP masters, it is possible to guarantee max latency for PL-based video.

5.2.3 DDR Port Arbitration

The PS interconnect uses all four QoS signals except where it attaches to the DDR memory controller, which takes only the most significant QoS signal. A 3-input mux selects among this QoS signal, another signal from the SLCR.DDR_URGENT register, and a DDRARB signal directly from the PL to determine if a request is urgent. Refer to the DDR memory controller chapter for more details.

5.3 AXI_HP Interfaces

The four AXI_HP interfaces provide PL bus masters with high bandwidth datapaths to the DDR and OCM memories. Each interface includes two FIFO buffers for read and write traffic. The PL to memory interconnect routes the high-speed AXI_HP ports to two DDR memory ports or the OCM. The AXI_HP interfaces are also referenced as AFI (AXI FIFO interface), to emphasize their buffering capabilities.

5.3.1 Features

The interfaces are designed to provide a high throughput data path between PL masters and PS memories including the DDR and on-chip RAM. The main features include:

- 32- or 64-bit data wide master interfaces (independently programmed per port)
- Efficient dynamic upsizing to 64-bits for aligned transfers in 32-bit interface mode, controllable via AxCACHE[1]
- Automatic expansion to 64-bits for unaligned 32-bit transfers in 32-bit interface mode
- Programmable release threshold of write commands
- Asynchronous clock frequency domain crossing for all AXI interfaces between the PL and PS
- Smoothing out of “long-latency” transfers using 1 KB (128 by 64 bit) data FIFOs for both reads and writes
- QoS signaling available from PL ports
- Command and Data FIFO fill-level counts available to the PL
- Standard AXI 3.0 interfaces supported
- Programmable command issuance to the interconnect, separately for read and write commands
- Large slave interface read acceptance capability in the range of 14 to 70 commands (burst length dependent)
- Large slave interface write acceptance capability in the range of 8 to 32 commands (burst length dependent)

5.3.2 Block Diagram

Figure 5-3 shows the block diagram for the AXI_HP interfaces.

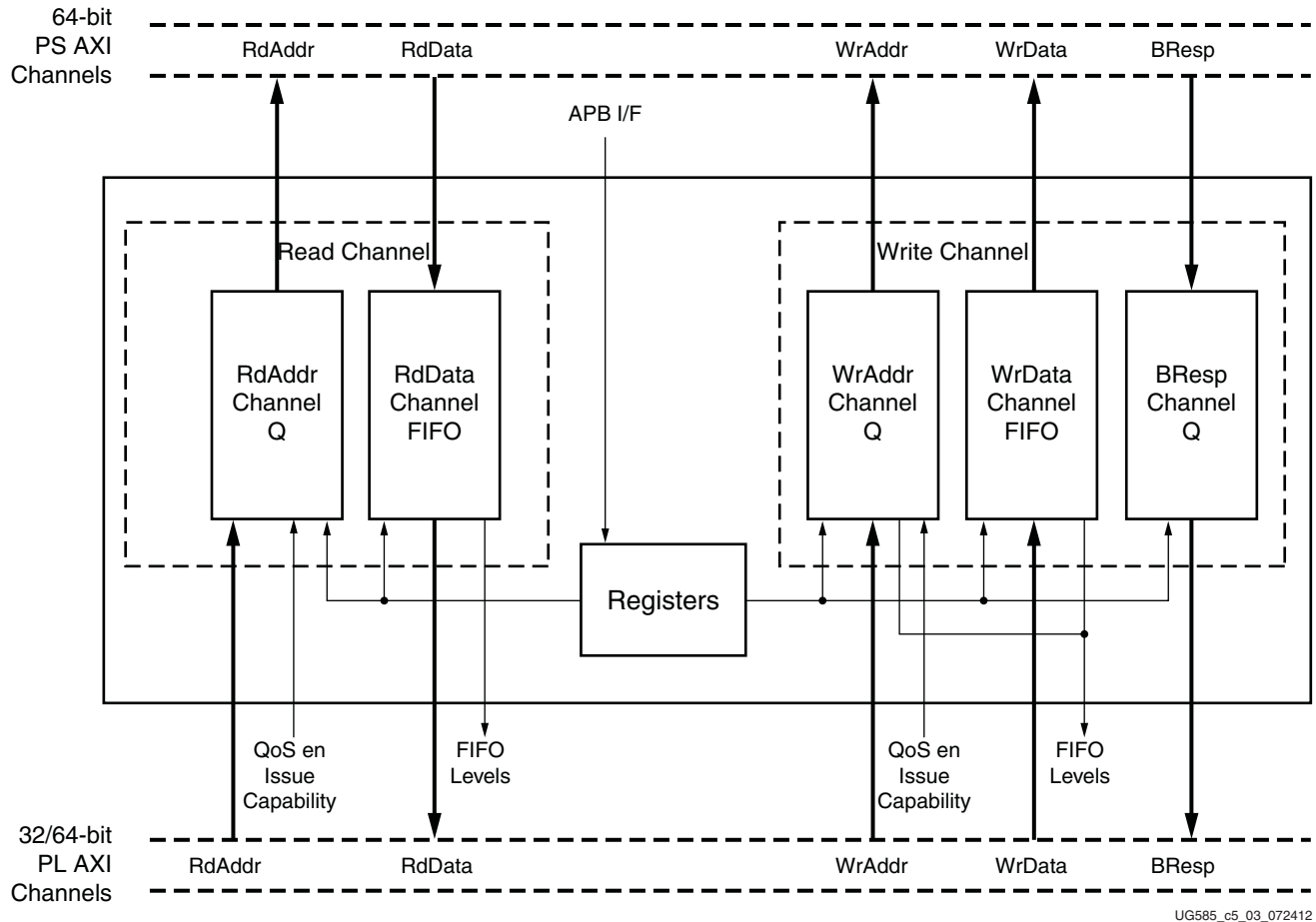


Figure 5-3: AXI_HP Block Diagram

5.3.3 Functional Description

There are two sets of AXI ports, one set connecting directly to the PL and the other connecting to the AXI interconnect matrix, allowing access to DDR and OCM memory (see [Figure 5-4](#)).

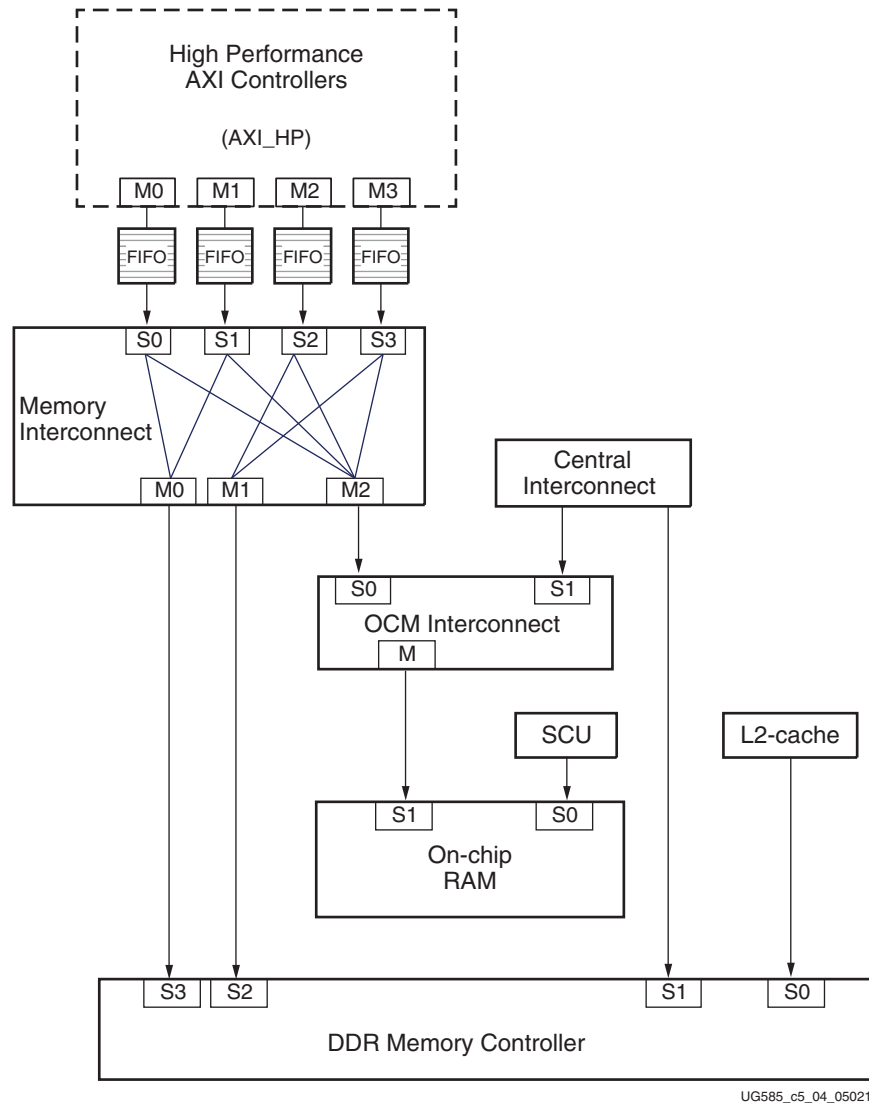


Figure 5-4: High Performance (AXI_HP) Connectivity

5.3.4 Performance

See [Chapter 22, Programmable Logic Design Guide](#) for more information.

5.3.5 Register Overview

A partial list of registers related to the high performance AXI port is listed in [Table 5-5](#)

Table 5-5: High Performance (AFI) AXI Register Overview

Module	Register Name	Overview
AXI_HP	AFI_RDCHAN_CTRL AFI_WRCHAN_CTRL	Select 64- or 32-bit interface width mode. Various bandwidth management control settings.
	AFI_RDCHAN_ISSUINGCAP AFI_WRCHAN_ISSUINGCAP	Maximum outstanding read/write commands.
	AFI_RDQOS AFI_WRQOS	Read/write register-based Quality of Service (QoS) priority value.
	AFI_RDDATAFIFO_LEVEL AFI_WRDATAFIFO_LEVEL	Read/write data FIFO register occupancy.
OCM	OCM_CONTROL	Change arbitration priority of HP (and central interconnect) accesses at OCM with respect to SCU writes.
DDRC	axi_priority_rd_port2 axi_priority_wr_port2	Various priority settings for arbitration at DDR controller for AXI_HP (AFI) ports 2 and 3.
	axi_priority_rd_port3 axi_priority_wr_port3	Various priority settings for arbitration at DDR controller for AXI_HP (AFI) ports 0 and 1.
SLCR	LVL_SHFTR_EN	Level shifters. Must be enabled before using any of the PL AXI interfaces.

5.3.6 Bandwidth Management Features

For applications requiring multiple programmable logic masters on multiple high performance AXI interface ports simultaneously, and in the presence of a medium or heavily loaded PS system, the management of the bandwidth per programmable logic port or “thread” becomes more difficult.

For example, if real-time type traffic is required on one thread, possibly mixed with non real-time traffic on other threads/ports, the standard AXI 3.0 bus protocol does not explicitly provide methods to manage priority.

The high performance AXI interface module does provide several functions to assist priority and queue management. The majority of management functions are provided to both the programmable logic design as PL signals and the PS as registers, as performance optimization is application dependent. This allows maximum flexibility, while simplifying the high performance AXI interface requirements.

The additional signals provided to the PL, in addition to standard AXI3 signals are provided in [Table 5-6](#). The priority and occupancy management functions provided are discussed in the following sections.

Table 5-6: Additional per-port HP PL Signals

Type	PS-PL Signal Name	I/O	Description
FIFO Occupancy	SAXIHP{0-3}RCOUNT[7:0]	O	Fill level of the RdData Channel FIFO
	SAXIHP{0-3}WCOUNT[7:0]	O	Fill level of the WrData Channel FIFO
	SAXIHP{0-3}RACOUNT[2:0]	O	Fill level of the RdAddr Channel FIFO
	SAXIHP{0-3}WACOUNT[5:0]	O	Fill level of the WrAddr Channel FIFO
Quality of Service	SAXIHP{0-3}AWQOS[3:0]	I	WrAddr Channel QOS input. Qualified by SAXIHP{0-3}AWVALID
	SAXIHP{0-3}ARQOS[3:0]	I	RdAddr Channel QOS input. Qualified by SAXIHP{0-3}ARVALID
Interconnect Issuance Throttling	SAXIHP{0-3}RDISSUECAP1EN	I	When asserted (1), indicates that the Maximum Outstanding Read Commands (Issuing Capability) should be derived from the "rdIssueCap1" register.
	SAXIHP{0-3}WRISSUECAP1EN	I	When asserted (1), indicates that the Maximum Outstanding Write Commands (Issuing Capability) should be derived from the "wrIssueCap1" register.

QoS Priority

The AXI QoS input signals can be used to assign an arbitration priority to the read and write commands.

Note that the PS interconnect allows either master control or programmable (register) control as a configuration option. For the AFI, it is desirable to have the ability for masters to dynamically change the QOS inputs. However, to provide flexibility, the register field AFI_RDCHAN_CTRL.FabricQosEn is provided. This allows a static QoS value to be programmed via the high performance AXI interface port, ignoring the PL AXI QoS inputs.

FIFO Occupancy

The level of the data and command FIFOs for both read and write are exported to the PL allowing the user to take advantage of the QOS feature supported by the top-level interconnect. Based on the relative levels of these FIFOs, a PL controller could dynamically change the priority of the individual read and write requests into the high performance AXI interface block(s). If, for instance, a particular PL master read data FIFO is getting too empty, the priority of the read requests could be increased.

The filling of this FIFO now takes priority over the other three FIFOs. When the FIFO reaches an acceptable fill-level, the priority typically is reduced again. The exact scheme used to control the relative priorities is flexible as it must be performed in the programmable logic. Note that the "FIFO Level" should be used as a relative level as opposed to an exact level, because clock domain crossing is involved.

Another possible application of the FIFO levels is using them to "look-ahead" at the data fill level to determine if data can be read or written without having to use the AXI RVALID/WREADY handshake signals. This could potentially simplify the AXI interface design logic, enabling higher speeds of operation.

Interconnect Issuance Throttling

To optimize the latency or throughput of other masters in the system such as the CPUs, it might be desirable to constrain the number of outstanding transactions that a high-performance port requests to the system interconnect. Issuing capability is the maximum number of outstanding commands that a HP can request at any one time.

Control of the read and write command issuing capability of the high performance AXI interface is available as a primary input from the logic. This option can be enabled via the DDRC.AFI_XXCHAN_CTRL FabricOutCmdEn registers.

The logic signals, SAXIHP{0-3}RDISSUECAP1_EN and SAXIHP{0-3}WRISSUECAP1_EN allow the user to change the issuing capability of the AFI block to the PS dynamically between two levels.

Write FIFO Store and Forward

The write channels can be configured to store-and-forward write commands or allow them to pass through with no storage. The following two registers control the mode of write store and forward:

- AFI_WRCHAN_CTRL.[WrCmdReleaseMode]
- AFI_WRCHAN_CTRL.[WrDataThreshold]

The Mode register selects between a complete AXI burst store-and-forward, a partial AXI burst store-and-forward, or a pass through (no store at all). If absolute minimum latency for write commands is required, the pass-through mode could be selected. However, in cases where multiple masters are competing for the system slaves, better system performance can likely be achieved using at least the partial AXI burst store-and-forward mode. This is because once an AXI write is committed at each point throughout the PS, the entire burst must be processed before any other write data from other write commands can be processed.

For example, using pass-through mode, if one HP port with a slow clock rate issues a long burst, a second port with a faster clock rate might need to wait until the entire slower write data burst has been transferred, even if all of the write data on the fast clock is available. This is different from the case of reads, where read data interleaving is permitted.

32-bit Interface Considerations

Each physical high performance PL AXI interface is programmable to be either a 32-bit or 64-bit interface through the register field AFI.AFI_xxCHAN_CTRL.[32BitEn]. Note that the read and write channels have separate enables and can therefore be configured differently.

Upsizing and Expansion

In 32-bit mode, some form of translation between the 32-bit port and the 64-bit port is required. For write data, the 32-bit data (and write strobes) must be aligned correctly onto the appropriate lanes in the 64-bit domain. For read data, the appropriate lanes of the 64-bit data must be aligned onto the 32-bit data bus. This data alignment between different width interfaces were automatically dealt with by the high performance AXI interface module.

For the 32-bit mode, an “expansion” or “upsizing” must be performed to the 64-bit bus. These are defined as follows:

- Expansion. The AxSIZE[] and AxLEN[] fields remain unchanged on the 64-bit bus. The number of data beats in the 64-bit domain is therefore the same as the number of data beats in the 32-bit domain. This is the simplest option but also the most inefficient in terms of bandwidth utilization.
- Upsizing. This is an optimization that makes better use of the 64-bit bus available bandwidth. The AxSIZE[] field can be changed to `64-BIT (expansion case it is `32-BIT or less) and the AxLEN[] field can potentially be adjusted to make use of the 64-bit bus. For a full width transfer, the number of data beats in the 64-bit domain is now, at best, *half* the number of data beats in the 32-bit domain. For example, a burst of 16x32-bit is upsized to a burst of 8x64-bit.

Note: Upsizing only occurs if the AxCACHE[1] bit is set. If it is not, expansion of the command occurs. This means that the user can dynamically control, on a per-command basis, whether to expand or upsize.

Note: In 64-bit mode, there is no translation between the programmable logic transactions and the internal 64-bit PS transactions. Whatever appears at the PL port is passed as is to the PS port. In 64-bit mode no upsizing or expansion is performed. This also applies to narrow transactions in the 64-bit mode.

32-bit Interface Limitations

The high performance AXI interface imposes the following constraints:

1. In 32-bit mode, only burst multiples of 2, incremental burst read commands, aligned to 64-bit boundaries are upsized. All other 32-bit commands are expanded. These include all narrow transactions (wrap as well as fixed burst types).
2. Whenever an expanded read command is accepted from the programmable logic by the AFI, this command is blocked until all outstanding high performance AXI interface read commands in the pipeline are flushed. The flushing occurs automatically under control of the AFI.

The implication of the above is that for expanded commands, performance is very limited, as command pipelining is essentially disabled.

Note: All valid AXI command are still supported, just not optimized to take advantage of the 64-bit bus bandwidth.

In the case of write commands completing out-of-order, no performance penalty is incurred because the BRESP can be issued in any order directly back to the PL ports.

To be symmetric across read and write operations, the high performance AXI interface also only upsizes 64-bit aligned burst multiples of 2, incremental burst write commands, in 32-bit mode. In the case of writes however, no blocking” of expanded commands occur. Write performance for expanded commands in 32-bit mode is therefore much higher than read performance.

5.3.7 Transaction Types

Table 5-7 summarizes the different command types issued to the high performance AXI interface from the PL and the command modifications that occur.

Table 5-7: High Performance AXI Interface Command Types

No.	Mode	Command Type	Translation	Comments
1	64-bit	64-bit Reads All Burst types	None	Best optimization possible.
2	64-bit	Narrow Read	None	Because no upsizing is performed, the narrower the width, the more inefficient the transaction.
3	64-bit	64-bit Write All Burst types	None	Best optimization possible.
4	64-bit	Narrow Write	None	Because no upsizing is performed, the narrower the width, the more inefficient the transaction.
5	32-bit	32-bit INCR Read Aligned to 64-bits Even burst multiples	Upsized to 64-bits	Best 32-bit mode optimization possible.
6	32-bit	All other 32-bit Read Commands	Expanded to 64-bits	Each read command is blocked until all previous read commands are completed. Extremely inefficient.
7	32-bit	32-bit INCR Write Aligned to 64-bits Even burst multiples	Upsized to 64-bits	Best 32-bit mode optimization possible.
8	32-bit	All other 32-bit Write Commands	Expanded to 64-bits	Relatively inefficient because no upsizing is performed. No blocking occurs for writes.

5.3.8 Command Interleaving and Re-Ordering

When multi-threaded commands are used in AXI, there is the potential for commands being processed out-of-order as well as interleaving of data beats.

The DDR controller guarantees that all read commands are completed continuously, that is, it does not interleave read data onto its external AXI ports. It does however take advantage of re-ordering of read and write commands, to perform internal optimizations. It is therefore expected that read and write commands issued to the DDR controller are sometimes completed in a different order from which they were issued.

Read data interleaving is not supported by either the DDR or the OCM. However, the interconnect might introduce read data interleaving into the system when a single PL port issues multi-threaded read commands to both the DDR and OCM memories.

From the high performance AXI interface's perspective:

- Both read and write commands can be re-ordered.
- Read data interleaving might occur.

5.3.9 Performance Optimization Summary

This section summarizes the most important considerations when using the high performance AXI interface module from a software or user perspective.

- For general purpose AXI transfers, use the general purpose PS AXI ports and not these ports. These ports are optimized for high throughput applications, but have various limitations.
- [Table 5-5](#) summarizes the different command types issued to the high performance AXI interface module from the PL and the way in which they are dealt with.
- When using 32-bit mode, it is strongly recommended not to use commands of the type shown in line 6 of [Table 5-5](#), as this impacts performance significantly.
- The QoS PL inputs can be controlled from physical programmable logic signals or statically configured in APB registers. The signals allow QoS values to be changed on a per-command basis. The register control is static for all commands.
- The AxCACHE[1] must be set for upsizing to occur. If this bit is not set, expansion always occurs.
- If the PL design demands a continuous read data flow after the first data beat has been read, the design must first allow the read data FIFO to fill with the complete transaction data before popping the first data beat out. The FIFO level is exported to the PL for this purpose. This behavior might be useful if the PL master is not able to be throttled by RVALID after the first data exits the read FIFO.
- Wait-states can be inserted if write commands are not asserted at least one cycle ahead of the corresponding first write data beat in 32-bit AXI channel slave interface mode.
- The PL masters should be able to handle read data interleaving. If it is desired that they not deal with this issue, they should not issue multi-threaded read commands to both the OCM and DDR from the same port by using the same ARID value for all outstanding read requests.
- The relationship of write FIFO occupancy to the write data ready to accept signal (WREADY) varies as follows:
 - In 64-bit AXI mode: FIFO not full (SAXIHP0WCOUNT < 128) always implies WREADY=1.
 - In 32-bit AXI mode: There is a dependency between the write address (AWVALID) and the write data (WVALID). If the write address is presented at least one cycle before the first beat of any given write data burst, then the FIFO not full (SAXIHP0WCOUNT < 128) implies WREADY=1. If not, then WREADY is de-asserted until the write address is produced. The reason for this back pressure is that in 32-bit mode, expansion/upsizing is performed on the data into the write data FIFO.
- Write response (BVALID) latency is dependent on many factors, such as DDR latency, DDR transaction reordering, and other conflicting traffic (including higher-priority transactions). Write commands and data are sent the entire path to the slave (DDR or OCM) and the response is issued by the slave to return to the high performance AXI interface. Transactions issued after reception of the write response is guaranteed to be committed later at the slave than the responded write transaction.

5.4 AXI_ACP Interface

The accelerator coherency port provides low-latency access to programmable logic masters, with optional coherency with L1 and L2 cache. From a system perspective, the ACP interface has similar connectivity as the APU CPUs. Due to this close connectivity, the ACP directly competes with them for resource access outside of the APU block. [Figure 5-5](#) gives an overview of the ACP connectivity. The PL level shifters must be enabled via LVL_SHFTR_EN before PL logic communication can occur. For more information about the ACP interface, see [Chapter 3, Application Processing Unit](#).

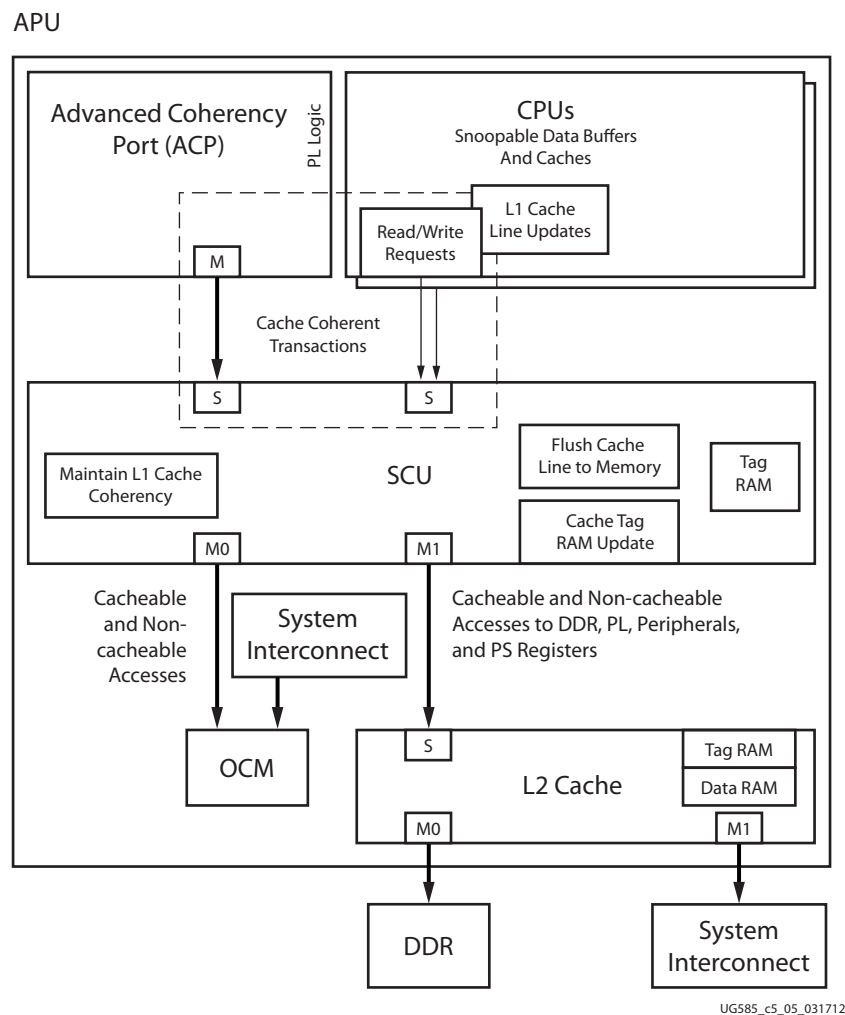


Figure 5-5: ACP Connectivity Diagram

5.5 AXI_GP Interfaces

5.5.1 Features

AXI_GP features include:

- Standard AXI protocol
- Data bus width: 32
- Master port ID width: 12
- Master port issuing capability: 8 reads, 8 writes
- Slave port ID width: 6
- Slave port acceptance capability: 8 reads, 8 writes

5.5.2 Performance

These interfaces are connected directly to the ports of the master interconnect and the slave interconnect, without any additional FIFO buffering, unlike the AXI_HP interfaces which has elaborate FIFO buffering to increase performance and throughput. Therefore, the performance is constrained by the ports of the master interconnect and the slave interconnect. These interfaces are for general-purpose use only and are not intended to achieve high performance.

5.6 Signals

5.6.1 AXI Signals

AXI signals are identified in [Table 5-8](#). The PL level shifters must be enabled via the LVL_SHFTR_EN register before PL logic communication can occur.

Table 5-8: AXI Signals Summary

AXI Channel	AXI PS Masters		AXI PS Slaves	
	M_AXI_GP{0,1}	I/O	S_AXI_GP{0,1} S_AXI_HP{0:3} S_AXI_ACP	I/O
Clock, Reset				
	MAXIGP{0,1}ACLK	I	SAXIGP{0,1}ACLK SAXIHP{0:3}ACLK SAXIACPACLK	I
	MAXIGP{0,1}ARESETN	O	SAXIGP{0,1}ARESETN SAXIHP{0:3}ARESETN SAXIACPARESETN	O

Table 5-8: AXI Signals Summary (Cont'd)

AXI Channel	AXI PS Masters		AXI PS Slaves	
	M_AXI_GP{0,1}	I/O	S_AXI_GP{0,1} S_AXI_HP{0:3} S_AXI_ACP	I/O
Read Address				
	MAXIGP{0,1}ARADDR[31:0]	O	SAXIGP{0,1}ARADDR[31:0] SAXIHP{0:3}ARADDR[31:0] SAXIACPARADDR[31:0]	I
	MAXIGP{0,1}ARVALID	O	SAXIGP{0,1}ARVALID SAXIHP{0:3}ARVALID SAXIACPARVALID	I
	MAXIGP{0,1}ARREADY	I	SAXIGP{0,1}ARREADY SAXIHP{0:3}ARREADY SAXIACPARREADY	O
	MAXIGP{0,1}ARID[11:0]	O	SAXIGP{0,1}ARID[5:0] SAXIHP{0:3}ARID[5:0] SAXIACPARID[2:0]	I
	MAXIGP{0,1}ARLOCK[1:0]	O	SAXIGP{0,1}ARLOCK[1:0] SAXIHP{0:3}ARLOCK[1:0] SAXIACPARLOCK[1:0]	I
	MAXIGP{0,1}ARCACHE[3:0]	O	SAXIGP{0,1}ARCACHE[3:0] SAXIHP{0:3}ARCACHE[3:0] SAXIACPARCACHE[3:0]	I
	MAXIGP{0,1}ARPROT[2:0]	O	SAXIGP{0,1}ARPROT[2:0] SAXIHP{0:3}ARPROT[2:0] SAXIACPARPROT[2:0]	I
	MAXIGP{0,1}ARLEN[3:0]	O	SAXIGP{0,1}ARLEN[3:0] SAXIHP{0:3}ARLEN[3:0] SAXIACPARLEN[3:0]	I
	MAXIGP{0,1}ARSIZE[1:0]	O	SAXIGP{0,1}ARSIZE[1:0] SAXIHP{0:3}ARSIZE[2:0] SAXIACPARSIZE[2:0]	I
	MAXIGP{0,1}ARBURST[1:0]	O	SAXIGP{0,1}ARBURST[1:0] SAXIHP{0:3}ARBURST[1:0] SAXIACPARBURST[1:0]	I
	MAXIGP{0,1}ARQOS[3:0]	O	SAXIGP{0,1}ARQOS[3:0] SAXIHP{0:3}ARQOS[3:0] SAXIACPARQOS[3:0]	I
	~		~ ~ SAXIACPARUSER[4:0]	I
Read Data				

Table 5-8: AXI Signals Summary (Cont'd)

AXI Channel	AXI PS Masters		AXI PS Slaves	
	M_AXI_GP{0,1}	I/O	S_AXI_GP{0,1} S_AXI_HP{0:3} S_AXI_ACP	I/O
	MAXIGP{0,1}RDATA[31:0]	I	SAXIGP{0,1}RDATA[31:0] SAXIHP{0:3}RDATA[63:0] SAXIACPRDATA[63:0]	O
	MAXIGP{0,1}RVALID	I	SAXIGP{0,1}RVALID SAXIHP{0:3}RVALID SAXIACPRVALID	O
	MAXIGP{0,1}RREADY	O	SAXIGP{0,1}RREADY SAXIHP{0:3}RREADY SAXIACPRREADY	I
	MAXIGP{0,1}RID[11:0]	I	SAXIGP{0,1}RID[5:0] SAXIHP{0:3}RID[5:0] SAXIACPRID[2:0]	O
	MAXIGP{0,1}RLAST	I	SAXIGP{0,1}RLAST SAXIHP{0:3}RLAST SAXIACPRLAST	O
	MAXIGP{0,1}RRESP[1:0]	I	SAXIGP{0,1}RRESP[2:0] SAXIHP{0:3}RRESP[2:0] SAXIACPRRESP[2:0]	O
	~		~ SAXIHP{0:3}RCOUNT[7:0] ~	O
	~		~ SAXIHP{0:3}RACOUNT[2:0] ~	O
	~		~ SAXIHP{0:3}RDISSEUECAP1EN ~	I
Write Address				
	MAXIGP{0,1}AWADDR[31:0]	O	SAXIGP{0,1}AWADDR[31:0] SAXIHP{0:3}AWADDR[31:0] SAXIACPAWADDR[31:0]	I
	MAXIGP{0,1}AWVALID	O	SAXIGP{0,1}AWVALID SAXIHP{0:3}AWVALID SAXIACPAWVALID	I
	MAXIGP{0,1}AWREADY	I	SAXIGP{0,1}AWREADY SAXIHP{0:3}AWREADY SAXIACPAWREADY	O
	MAXIGP{0,1}AWID[11:0]	O	SAXIGP{0,1}AWID[5:0] SAXIHP{0:3}AWID[5:0] SAXIACPAWID[2:0]	I

Table 5-8: AXI Signals Summary (Cont'd)

AXI Channel	AXI PS Masters		AXI PS Slaves	
	M_AXI_GP{0,1}	I/O	S_AXI_GP{0,1} S_AXI_HP{0:3} S_AXI_ACP	I/O
	MAXIGP{0,1}AWLOCK[1:0]	O	SAXIGP{0,1}AWLOCK[1:0] SAXIHP{0:3}AWLOCK[1:0] SAXIACPAWLOCK[1:0]	I
	MAXIGP{0,1}AWCACHE[3:0]	O	SAXIGP{0,1}AWCACHE[3:0] SAXIHP{0:3}AWCACHE[3:0] SAXIACPAWCACHE[3:0]	I
	MAXIGP{0,1}AWPROT[2:0]	O	SAXIGP{0,1}AWPROT[2:0] SAXIHP{0:3}AWPROT[2:0] SAXIACPAWPROT[2:0]	I
	MAXIGP{0,1}AWLEN[3:0]	O	SAXIGP{0,1}AWLEN[3:0] SAXIHP{0:3}AWLEN[3:0] SAXIACPAWLEN[3:0]	I
	MAXIGP{0,1}AWSIZE[1:0]	O	SAXIGP{0,1}AWSIZE[1:0] SAXIHP{0:3}AWSIZE[2:0] SAXIACPAWSIZE[2:0]	I
	MAXIGP{0,1}AWBURST[1:0]	O	SAXIGP{0,1}AWBURST[1:0] SAXIHP{0:3}AWBURST[1:0] SAXIACPAWBURST[1:0]	I
	MAXIGP{0,1}AWQOS[3:0]	O	SAXIGP{0,1}AWQOS[3:0] SAXIHP{0:3}AWQOS[3:0] SAXIACPAWQOS[3:0]	I
	~		~ ~ SAXIACPAWUSER[4:0]	I
Write Data				
	MAXIGP{0,1}WDATA[31:0]	O	SAXIGP{0,1}WDATA[31:0] SAXIHP{0:3}WDATA[63:0] SAXIACPWDATA[63:0]	I
	MAXIGP{0,1}WVALID	O	SAXIGP{0,1}WVALID SAXIHP{0:3}WVALID SAXIACPWVALID	I
	MAXIGP{0,1}WREADY	I	SAXIGP{0,1}WREADY SAXIHP{0:3}WREADY SAXIACPWREADY	O
	MAXIGP{0,1}WID[11:0]	O	SAXIGP{0,1}WID[5:0] SAXIHP{0:3}WID[5:0] SAXIACPWID[2:0]	I
	MAXIGP{0,1}WLAST	O	SAXIGP{0,1}WLAST SAXIHP{0:3}WLAST SAXIACPWLAST	I

Table 5-8: AXI Signals Summary (Cont'd)

AXI Channel	AXI PS Masters		AXI PS Slaves	
	M_AXI_GP{0,1}	I/O	S_AXI_GP{0,1} S_AXI_HP{0:3} S_AXI_ACP	I/O
	MAXIGP{0,1}WSTRB[3:0]	O	SAXIGP{0,1}WSTRB[3:0] SAXIHP{0:3}WSTRB[7:0] SAXIACPWSTRB[7:0]	I
	~		~ SAXIHP{0:3}WCOUNT[7:0] ~	O
	~		~ SAXIHP{0:3}WACOUNT[5:0] ~	O
	~		~ SAXIHP{0:3}WRISSUECAP1EN ~	I
Write Response				
	MAXIGP{0,1}BVALID	I	SAXIGP{0,1}BVALID SAXIHP{0:3}BVALID SAXIACPBVALID	O
	MAXIGP{0,1}BREADY	O	SAXIGP{0,1}BREADY SAXIHP{0:3}BREADY SAXIACPBREADY	I
	MAXIGP{0,1}BID[11:0]	I	SAXIGP{0,1}BID[5:0] SAXIHP{0:3}BID[5:0] SAXIACPBID[2:0]	O
	MAXIGP{0,1}BRESP[1:0]	I	SAXIGP{0,1}BRESP[1:0] SAXIHP{0:3}BRESP[1:0] SAXIACPBRESP[1:0]	O

5.6.2 Clocks and Resets

Each interface has a single clock for all five channels that make up an interface. This clock is provided by the PL. Each interface has a common reset signal driven by the PS reset subsystem.

All clocks must be active and all resets must be inactive on all PS-PL AXI interfaces for the GPV to function properly. The entire PS might hang if this condition is not satisfied and a GPV access is attempted. Therefore, no GPV access should be attempted if not all of the clocks for the PS-PL AXI interfaces are connected.

Boot and Configuration

6.1 Introduction

The Zynq-7000 EPP devices takes advantage of the on-chip CPU to facilitate configuration. Initial device configuration of the processing system (PS) and the programmable logic (PL) must take place through the PS when not using JTAG.

There are two major blocks that control configuration. The first is the BootROM which is a static block of memory that is executed by the MPCore after power-on reset and warm reset. The second major block is the device configuration unit which controls JTAG debug access and provides an interface to the AES, HMAC, and PCAP blocks for PL configuration and data decryption.

This chapter describes the following topics:

- [External Startup Requirements](#)

This covers device requirements for startup including power, resets, mode pins, and clocks.

- [BootROM](#)

This covers CPU controlled device startup.

- [Device Configuration Interface: PCAP/HMAC/AES](#)

Device configuration includes all the methods and procedures for initializing and configuring the PS and PL. The device configuration unit (DevC) within the PS provides the means for initializing and configuring both the PS and the PL under software control.

Both the PS and PL can be configured under PS control either securely or non-securely. Configuration under external host control is also possible via JTAG. Unlike other Xilinx 7 series devices, Zynq-7000 EPP devices do not support initial PL controlled configuration.

Configuration on the Zynq-7000 EPP devices is a multi-step process. The configuration process involves a minimum of two 2 stages, but generally requires three stages.

The stages are as follows:

- Stage 0: Referred to as the BootROM, this stage controls initial device startup. The BootROM is non-modifiable code executed by the processor after power-on reset and warm reset.
- Stage 1: This is generally a first stage boot loader (FSBL), but it can be any user-controlled code. Please refer to UG821, *Zynq-7000 EPP Software Developers Guide* for details about FSBL.

- Stage 2: This is generally the user processing system design, but it could also be a second stage boot loader (SSBL). This stage is completely within user control and is not described further in this chapter. Please refer users to UG821, *Zynq-7000 EPP Software Developers Guide* for details about FSBL and stage 2 images.

6.2 External Startup Requirements

6.2.1 Introduction

Zynq-7000 EPP devices have power, clock, and reset requirements that must be met for successful BootROM execution. This section describes those requirements.

6.2.2 Power Requirements

The requirements for the PS and PL are shown in [Table 6-1](#). These power requirements are only for the BootROM. In stage 1 the PL requires power to configure. For further details on the required power rails refer to [Chapter 24, Power Management](#).

Table 6-1: PS and PL Power Requirements

Boot Option	Secure	Processing System Power	Programmable Logic Power
Configuration via NAND, NOR, SD, or Quad-SPI Note: Secure JTAG configuration is not supported	Yes	Required	Required
Configuration via NAND, NOR, SD, or Quad-SPI	No	Required	Not Required
Configuration via JTAG	No	Required	Required

6.2.3 Clock Requirements

PS_CLK must be present and within required operating ranges as specified in the EPP device's applicable data sheet. For more information on the PS_CLK refer to [Chapter 25, Clocks](#). PS_CLK must be present before PS_POR_B is de-asserted.

6.2.4 Reset Requirements

There are two main external reset sources that impact the execution of the BootROM. The resets are:

- PS_POR_B: This reset is used to hold the PS in reset until all PS power supplies are at the required voltage levels. It must be held Low through PS power-up. PS_POR_B should be generated by the power supply "power-good" signal.
- PS_SRST_B: This reset is used to force a system reset. It can be tied or pulled High, and can be High during the PS supply power ramps.

Both of these external resets result in the following device flow:

- If power-on reset (POR) is issued, the BOOT_MODE register is updated
- If PLL is enabled (not bypass), device waits for the PLL to lock
- Debug reset (for CPU and DAP) is de-asserted
- Device enters same path as for a warm reset (see below)

There are also internal warm reset signals that result in the BootROM executing again. When these reset sources are triggered the following steps occur in the device:

- BIST logic clears memories and buffers
- All functional clocks are stopped
- All CPU and peripheral clocks are de-asserted
- CPU0 start to execute from the boot ROM

6.2.5 Mode Pin Settings

Five mode pins, mode[4:0], are used to indicate the boot source, JTAG mode, and PLL bypass selection. Two voltage mode signals, vmode[1:0], are used to indicate the voltage mode of the multiplexed I/O banks. The mode[4:0] and vmode[1:0] signals should be connected using pull-up or pull-down 20 K Ω resistors. A pull-up resistor to V_{CCO_MIO0} specifies a 1 and a pull-down resistor to ground specifies a 0.

The vmode signals are used to set the individual MIO_PIN registers to the appropriate LVCMOS18 or LVCMOS33 IO standard for all MIO pins while the ROM is running. After the BootROM completes, the FSBL can change the I/O standard from LVCMOS to HSTL or LVTTL as needed. The vmode[0] pin is used to set the I/O standard on Bank 0 and vmode[1] is used to set the I/O standard on Bank 1.

The mode and vmode signals are sampled three PS_CLK clocks after the Low-to-High transition of the PS_POR_B reset pin. After being sampled on POR, the mode values are stored in the BOOT_MODE register in the SLCR. The vmode values are stored in GPIOB_DRVR_BIAS_CTRL.

The mode pins are MIO[6:2] and the vmode pins are MIO[8:7]. The pins are used as follows:

- MIO[2] selects the JTAG mode
- MIO[5:3] selects the boot mode
- MIO[6] enables the PLLs
- MIO[8:7] configures the I/O bank voltage

The PS boot mode selections are shown in [Table 6-2](#). The BOOT_MODE bits correspond to the associated register.

Table 6-2: Boot_Mode Strapping MIO Pins

	vmode[1]	vmode[0]	BOOT_MODE [4]	BOOT_MODE [0]	BOOT_MODE [2]	BOOT_MODE [1]	BOOT_MODE [3]
	MIO[8]	MIO[7]	MIO[6]	MIO[5]	MIO[4]	MIO[3]	MIO[2]
Cascaded JTAG							0
Independent JTAG							1
Boot Devices							
JTAG				0	0	0	0
NOR				0	0	1	
NAND				0	1	0	
Reserved				0	1	1	
Quad-SPI				1	0	0	
Reserved				1	0	1	
SD				1	1	0	
Reserved				1	1	1	
PLL Mode							
PLL Used			0				
PLL Bypassed			1				
MIO Bank 0 Voltage							
2.5 V, 3.3 V		0					
1.8 V		1					
MIO Bank 1 Voltage							
2.5 V, 3.3 V	0						
1.8 V	1						

6.3 BootROM

6.3.1 Introduction and Features

The BootROM provides the following configuration features:

- Three different PS configuration methods, including two master modes and one slave mode:
 - Secure, encrypted image, master mode
 - Non-secure master mode
 - Non-secure slave mode via JTAG

- Four master boot sources:
 - Quad-SPI flash
 - NAND flash
 - NOR flash
 - SD
- PS secure configuration using AES-256 and HMAC (SHA-256)
- SoC debug security
- Execute-in-place from NOR and QSPI

PS configuration starts after power-on reset. The ARM CPU starts executing code from the on-chip BootROM with JTAG disabled. The BootROM contains code for base drivers for NAND, NOR, Quad-SPI, SD, and PCAP. DDR and other peripheral initializations are not performed from the BootROM and must be done in the stage 1 image or later. For security reasons, the CPU is always the first device out of reset among all master modules within the PS. While the BootROM is running JTAG is disabled to ensure security.

The BootROM code is also responsible for loading the stage 1 boot image. EPP hardware supports multi-stage user boot image loading; any further user boot image loading after stage 1 is the user's responsibility. When the BootROM releases control to stage 1, user software assumes full control of entire system. The only way to execute the BootROM again is by performing a reset.

The PS boot source is selected via the mode-pin signals (indicated by a weak pull-up or pull-down applied to specific pins), which are sampled once during power-on reset. The sampled value is stored in the BOOT_MODE register.

The BootROM supports encrypted and unencrypted images referred to as *secure boot* and *non-secure boot*, respectively. Additionally the BootROM supports beginning execution of the stage 1 image from OCM after copying the image or executing direct from linear flash (NOR or QSPI) when using the execute-in-place (XIP) feature.

In secure boot the CPU, running from secure BootROM code, decrypts and authenticates the incoming user PS image, stores it in the OCM RAM, and then branches into it. In non-secure boot the CPU, running from BootROM code, disables all secure boot features including the AES engine within the PL before branching to the user image in the OCM RAM or flash, if XIP is used. The PS boot image is limited to 192 KB unless booting with XIP.

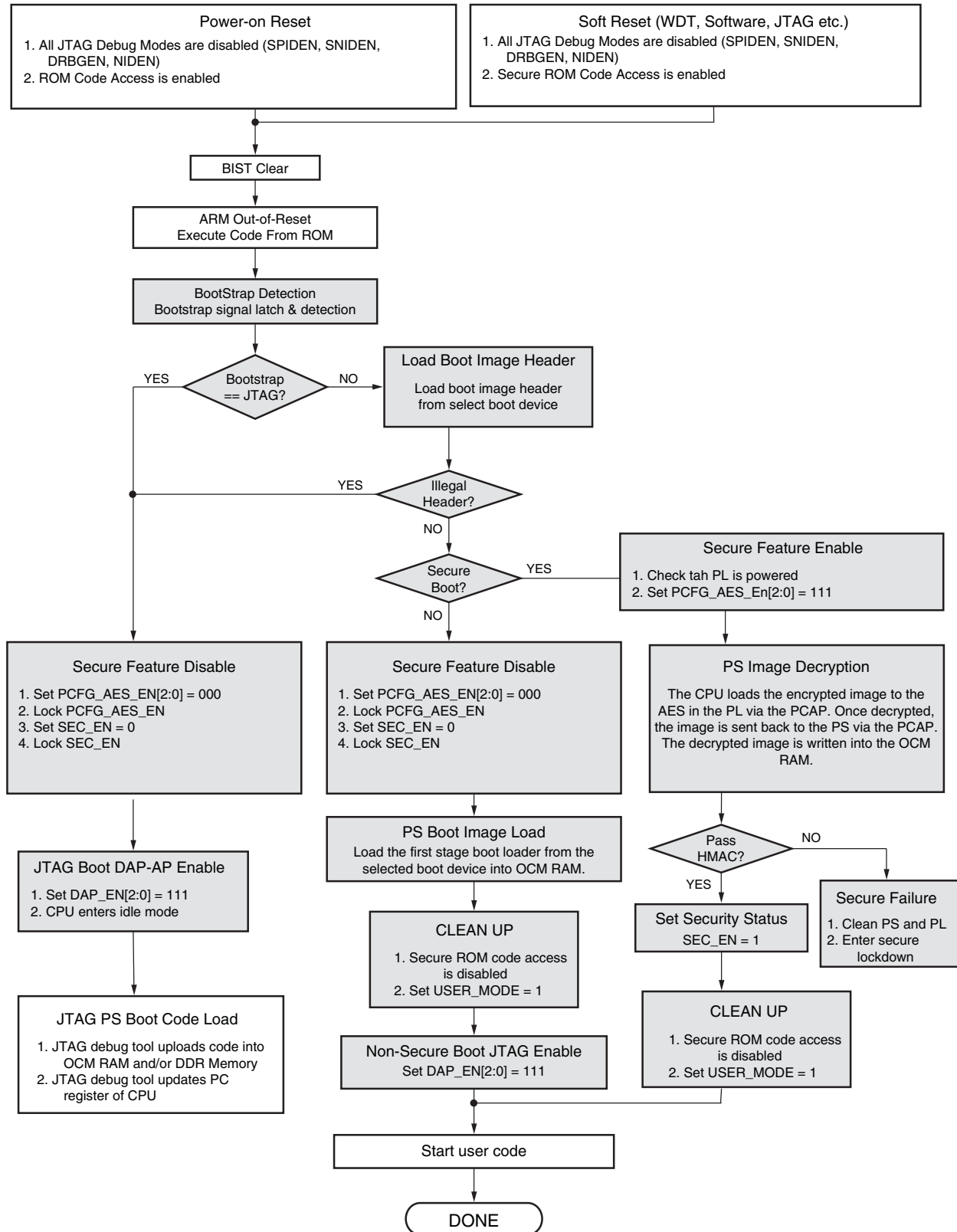
Any subsequent boot stages for either the PS or the PL are the user's responsibility and are under the user's control. The BootROM code is not accessible to the user. Following a stage 1 secure boot, the user can proceed with either secure or non-secure subsequent boot stages. Following a non-secure first stage boot, only non-secure subsequent stage boots are possible.

For decryption and authentication, the PS uses the hard-wired AES-256 and SHA-256 modules within the PL. For this reason, the PL must be powered up during any secure boot, even if only the PS is configured. The device encryption key is user-selectable from either the on-chip eFUSE unit or the on-chip BBRAM.

There are five possible boot sources: NAND, NOR, SD, Quad-SPI, and JTAG. The first four boot sources are used in master boot methods in which the CPU loads the external boot image from non-volatile memory into the PS.

JTAG can only be used in slave boot mode, and only supports a non-secure boot. An external host computer acts as the master to load the boot image into the OCM via a JTAG connection. The PS CPU remains in idle mode as the boot image is loaded.

The high-level configuration flow for the BootROM is shown in [Figure 6-1](#).



UG585_c6_01_071012

Figure 6-1: BootROM Configuration Flow

6.3.2 BootROM Header

The BootROM requires a header for all configuration interfaces except JTAG. The BootROM header format is shown in Table 6-3. For non-secure mode, the PS image is either retrieved from flash and loaded into on-chip RAM directly, or the processor jumps directly to the start location in the NOR/Quad-SPI flash memory. For secure mode, the PS image is retrieved from flash, decrypted, and authenticated using the AES and HMAC hardware within the PL.

The header controls these BootROM features:

- Execute-in-place (length and start of execution fields)
- Encryption key source (encryption status field)
- Dual Quad-SPI parallel configuration (width detection field)

Table 6-3: BootROM Header Format

Fields	Header Word Offset
Reserved for interrupts	0x00 - 0x01C
Width Detection	0x020
Image Identification	0x024
Encryption status	0x028
User Defined	0x02C
Source Offset	0x030
Length of Image	0x034
Reserved	0x038
Start of Execution	0x03C
Total Image Length	0x040
Reserved	0x044
Header Checksum	0x048
Reserved	0x04C - 0x09C
Register Initialization	0x0A0 - 0x89C
FSBL Image	0x8A0

Table 6-3 shows the layout of the flash boot ROM header with offsets relative from the beginning of the flash ROM. Each word is 32 bits in size. The header between 0 and 0x8A0 is always in clear non-encrypted form regardless of the state of the encryption status word.

Default values, usage, and behavior are described in the following sections.

Reserved for Interrupts — Byte Offset 0x0

Twenty-eight (28) words are reserved for interrupts. This is useful for execute-in-place for NOR and QSPI devices. It allows the vector table to be managed in two ways. The first is to use the MMU to remap the flash linear address space to 0x0. The second method to manage the vector table location is to use the coprocessor VBAR register. For more information on setting this register refer to the *ARM v7-AR Architecture Reference Manual*.

Width Detection — Byte Offset 0x20

This word has a mandatory value of 0xAA995566. This value allows the boot ROM to determine data width of a single Quad-SPI device and optional presence of a second parallel Quad-SPI device at power-up. Based on the data read, the BootROM reconfigures the flash interface so that the flash ROM can be read by the CPU as a linear address space. If this value cannot be found, the BootROM performs a lockdown.

Image Identification — Byte Offset 0x24

This word has a mandatory value of 0x584C4E58, 'XLNX'. This value allows the BootROM (along with the width detection word) to determine that a valid flash ROM boot header is present. If the value is not matched, the boot ROM does a lockdown.

Encryption Status — Byte Offset 0x28

If the boot image is encrypted valid values for this field are:

- 0xA5C3C5A3 (E-Fuse key source)
- 0x3A5C3C5A (Battery-backed RAM key source)

A value of 0 (zero) indicates a non-encrypted flash ROM boot image. Any other value causes the boot ROM to do a lockdown.

User Defined — Byte Offset 0x2C

This word is user-defined and can be any value. The boot ROM does not interpret or use this field.

Source Offset — Byte Offset 0x30

This word contains the number of bytes from the beginning of the flash image where the FSBL load image resides. This value is relative to the beginning of flash. The value must be at or above address offset 0x8C0 because the offset must be aligned to a 64-byte boundary (the lower six bits must be zero).

Length of Image — Byte Offset 0x34

This word contains the byte count of the load image to transfer to OCM memory. Transfers stop at the top of OCM memory regardless of any leftover count, causing a security lockup. The maximum image length is 0x30000.

A value of zero with NOR or QSPI flash causes the boot ROM to execute the FSBL from the associated flash device without copying the image to OCM (XIP).

Reserved — Byte Offset 0x38

This word region is reserved for future use and must be initialized to 0x0.

Start of Execution — Byte Offset 0x3C

This word has different valid values depending on whether execute-in-place is being used.

- If execute-in-place is not used:
 - This word must contain the start address relative to the OCM.
 - For non-secure mode the address must be equal to or greater than 0x0 and less than 0x30000.
 - For secure mode the address must equal 0x0.
 - Execution attempts outside of the OCM memory space cause a security lockup.
- If execute-in-place is used:
 - This word contains the start address within the Flash device (NOR or QSPI).
 - The address must be within the first 16 MB of the NOR or QSPI linear address space.

Total Image Length — Byte Offset 0x40

This word contains the total number of bytes loaded by the boot ROM from flash. For non-secure boot this field should equal "Length of Image". For secure images this field is larger than "Length of Image".

Reserved — Byte Offset 0x44

This word region is reserved for future use and must be initialized to 0x0.

Header Checksum — Byte Offset 0x48

The word contains the checksum value of the header which is checked prior to using the data within the header. The checksum is calculated by summing the words from 0x20 to 0x44 and inverting the result.

Reserved — Byte Offset 0x4C

This word region is reserved for future use and must be initialized to 0x0.

Register Initialization — Byte Offset 0xA0

This word region contains a simple system to initialize PS registers for the MIO multiplexer and flash clocks. This allows the MIO multiplexer to be fully configured before the stage 1 image is copied into OCM or executed from flash with XIP, and allows for flash device clocks to be set to maximum bandwidth speeds.

A register initialization appears as two words, first a register address, then a register value. Register initializations can be in any order, and the same register can be initialized with different values as many times as needed. The register initialization is performed prior to copying the stage 1 image allowing modification of the default system setup for maximum configuration performance.

The allowable address range depends on whether the image is secure or non-secure:

- Secure mode allows addresses from 0xF8000100 to 0xF80001B4
- Non-secure mode allows addresses from 0xE0000000 to 0xFFF00000

The BootROM stops register initialization when either 0xFFFFFFFF register address is encountered or after the last register initialization pair (0x898 - 0x89F). Register address values not equal to 0xFFFFFFFF or the allowed range cause the boot ROM to perform a security lookup.

FSBL Image — Byte Offset 0x8A0

The FSBL image must start at or above this location.

6.3.3 Boot Performance

Various factors determine the PS configuration time in the BootROM. Those factors include power supply ramp time, configuration mode selected, external non-volatile device used, PS boot image size, and whether the PL is powered. The BootROM does not configure the PL so additional boot time calculations are needed for the stage 1 image. This section only covers the startup time requirements for the BootROM to load an image to the OCM and begin execution.

This section makes the following assumptions:

- The boot timer starts when the user turns on the power switch.
- The boot timer stops when the BootROM branches to the stage 1 image.
- All times assume default system values.

The boot timeline is divided into five stages:

1. Power ramp time
2. PLL lock time (TPSPLLLOCK)
3. PL clear time (TPOR)
4. Register initialization
5. Stage 1 image copy/execution

Power supply ramp times can be long. For boot-time critical applications, the user should consider using a high-performance power supply to reduce power ramp time.

PLL lock time could vary from 1000 reference clock cycles to 10,000 reference clock cycles, but it is relatively small compared to the other boot stages. Additionally, register initialization takes a negligible amount of time, but can have a drastic effect on the stage 1 image copy when XIP is not used.

The PL clear time is variable depending on whether the PL is powered while the BootROM is running. Secure boot mode and JTAG boot mode require the PL to be powered while the BootROM is running. If the PL is powered while the BootROM is running the startup time is extended by TPOR.

Given the high internal bandwidth of the DevC DMA module (100 MB/s secure, 400 MB/s non-secure) image load time is typically limited by the non-volatile RAM bus bandwidth.

The stage 1 image copy and execute time varies greatly depending on the configuration interface. For specific numbers refer to the applicable Zynq 7000 data sheet.

As this is the default stage 1 startup time these numbers can be significantly reduced using the register initialization table in the BootROM header.

6.3.4 Boot Devices

The BootROM supports configuration from four different slave interfaces:

- Quad-SPI
- NAND
- NOR flash
- SD

Additionally Zynq-7000 devices can be programmed via a JTAG. This section details features of each configuration interface.

Quad-SPI Flash

Quad-SPI booting has these features:

- x1, x2, and x4 single device configuration
- Dual x4 device configuration
- Execute-in-place

The BootROM uses the width-detection word to determine whether configuration is taking place from a single Quad-SPI device or dual parallel Quad-SPI devices.

The BootROM starts Quad-SPI configuration by enabling the single device pins and then attempts to read the width-detection word. If it sees the expected value it continues with the appropriate x4, x2, or x1 setup. If the BootROM sees a width-detection word value of 0xFFFFFFFF then it stops processing and falls back to non-secure JTAG boot. If it does not see the expected value of 0xAA995566 and it does not see an erased value 0xFFFFFFFF then it tries dual x4 Quad-SPI configuration.

If configuration is taking place from a single Quad-SPI flash the pins in [Table 6-4](#) are enabled.

Table 6-4: Quad-SPI MIO Pins

Signal Name	I/O	MIO Pin
QSPI0_IO[3:0]	IO	MIO[5:2]
QSPI0_SCLK	O	MIO[6]

Table 6-4: Quad-SPI MIO Pins (Cont'd)

Signal Name	I/O	MIO Pin
QSPI_SCLK_FB_OUT (if running > 40 MHz)	IO	MIO[8]
QSPI0_SS_B	O	MIO[1]

If a dual x4 Quad-SPI configuration is used the pins in Table 6-5 are enabled in addition to the single device pins.

Table 6-5: Additional Dual x4 Quad-SPI MIO Pins

Signal Name	I/O	MIO Pin
QSPI1_IO[3:0]	IO	MIO[13:10]
QSPI1_SCLK	O	MIO[9]
QSPI1_SS_B	O	MIO[0]

The BootROM uses the linear addressing feature of the Quad-SPI controller so only the first 16 MB are accessible for storing the stage 1 image. Memory above the first 16 MB is available after the BootROM passes execution to the stage 1 image.

The BootROM sets QSPI.LQSPI_CFG to use these settings:

- CLK_POL: 0
- CLK_PH: 0
- BAUD_RATE_DIV: 1 (by 4)
- INST_CODE is set as:
 - In x1 mode: 0x03
 - In x2 mode: 0x3B
 - In x4 mode: 0x6B
- DUMMY_BYTE is set as:
 - In x1 mode: 0
 - In x2 and x4 mode: 1
- SEP_BUS and TWO_MEM are set if a dual x4 configuration is used

NAND

NAND boot has these features:

- 8-bit or 16-bit NAND flash devices
- Tested densities up to 8 Gb
- Supports ONFI 1.0 device protocol

The MIO pin assignments for 8- and 16-bit boot modes are listed in Table 6-6.

Table 6-6: NAND MIO Pins

Signal Name	I/O	MIO Pin
SMC_NAND_CS_B[0]	IO	MIO[0]
Reserved		MIO[1]
SMC_NAND_ALE	IO	MIO[2]
SMC_NAND_WE_B	O	MIO[3]
SMC_NAND_DATA[2]	IO	MIO[4]
SMC_NAND_DATA[0]	IO	MIO[5]
SMC_NAND_DATA[1]	IO	MIO[6]
SMC_NAND_CLE	O	MIO[7]
SMC_NAND_RD_B	O	MIO[8]
SMC_NAND_DATA[4:7]	IO	MIO[9:12]
SMC_NAND_DATA[3]	IO	MIO[13]
SMC_NAND_BUSY	I	MIO[14]
SMC_NAND_DATA[15:8] ⁽¹⁾	IO	MIO[23:16]

1. The BootROM reads the ONFI compliant parameter information in 8-bit mode to determine the device width. If the device is 16 bits wide, then the BootROM enables these upper eight I/O signals.

Note: The 16-bit NAND interface is not available in the 7z010 CLG225 device.

The BootROM uses the following NAND timing values:

- $t_{rr} = 0 \times 02$
- $t_{ar} = 0 \times 02$
- $t_{clr} = 0 \times 01$
- $t_{wp} = 0 \times 03$
- $t_{rea} = 0 \times 02$
- $t_{wc} = 0 \times 05$
- $t_{rc} = 0 \times 05$

NOR

NOR boot has these features:

- Execute-in-place
- Support for asynchronous flash
- Supports x8 flash devices
- Densities up to 256 Mb

The BootROM does not try to perform any configuration detection of NOR flash devices. When NOR is selected for configuration the BootROM enables the pins in [Table 6-7](#).

Note: The NOR and SRAM interfaces are not available in the 7z010 CLG225 device.

Table 6-7: NOR MIO Pins

Signal Name	I/O	MIO Pin
SMC_SRAM_CS_B[1:0]	O	MIO[1:0]
Reserved	O	MIO[2]
SMC_SRAM_DQ[3:0]	I/O	MIO[6:3]
SMC_SRAM_OE_B	O	MIO[7]
SMC_SRAM_WE_B	O	MIO[8]
SMC_SRAM_DQ[7:6]	I/O	MIO[10:9]
SMC_SRAM_DQ[4]	I/O	MIO[11]
SMC_SRAM_WAIT	I	MIO[12]
SMC_SRAM_DQ[5]	I/O	MIO[13]
Reserved	I	MIO[14]
SMC_SRAM_ADDR[24:0]	O	MIO[39:15]

The shared memory controller is set up to operate with the following timing parameters in the BootROM (Register SMC.SET_CYCLES):

- $t_{rc} = 7$
- $t_{wc} = 7$
- $t_{ceoe} = 2$
- $r_{wp} = 5$
- $t_{pc} = 2$
- $t_{tr} = 1$
- $we_time = 0$

JTAG

When the JTAG boot source is selected, the CPU halts immediately after it disables access to all security-related items and enables the JTAG port. It is the user's responsibility to manage downloading the boot image into OCM RAM or DDR memory via JTAG before waking up the CPU and continuing the boot process.

SD

SD boot supports these features:

- Boot from standard SD or SDHC cards
- FAT32 file system support for storing initial image
- Up to 32 GB densities

Note: The SD card boot mode is not supported in the 7z010 CLG225 device.

The BootROM performs these steps while booting SD:

1. Initializes the MIO pins listed in [Table 6-8](#)
2. Sets the SD clock to run at ~226 KHz with a 33.333 MHz PS_CLK
3. Reads BOOT.BIN from the root of the SD file system and copies it into OCM after parsing the required BootROM header
4. Starts execution from the beginning of OCM

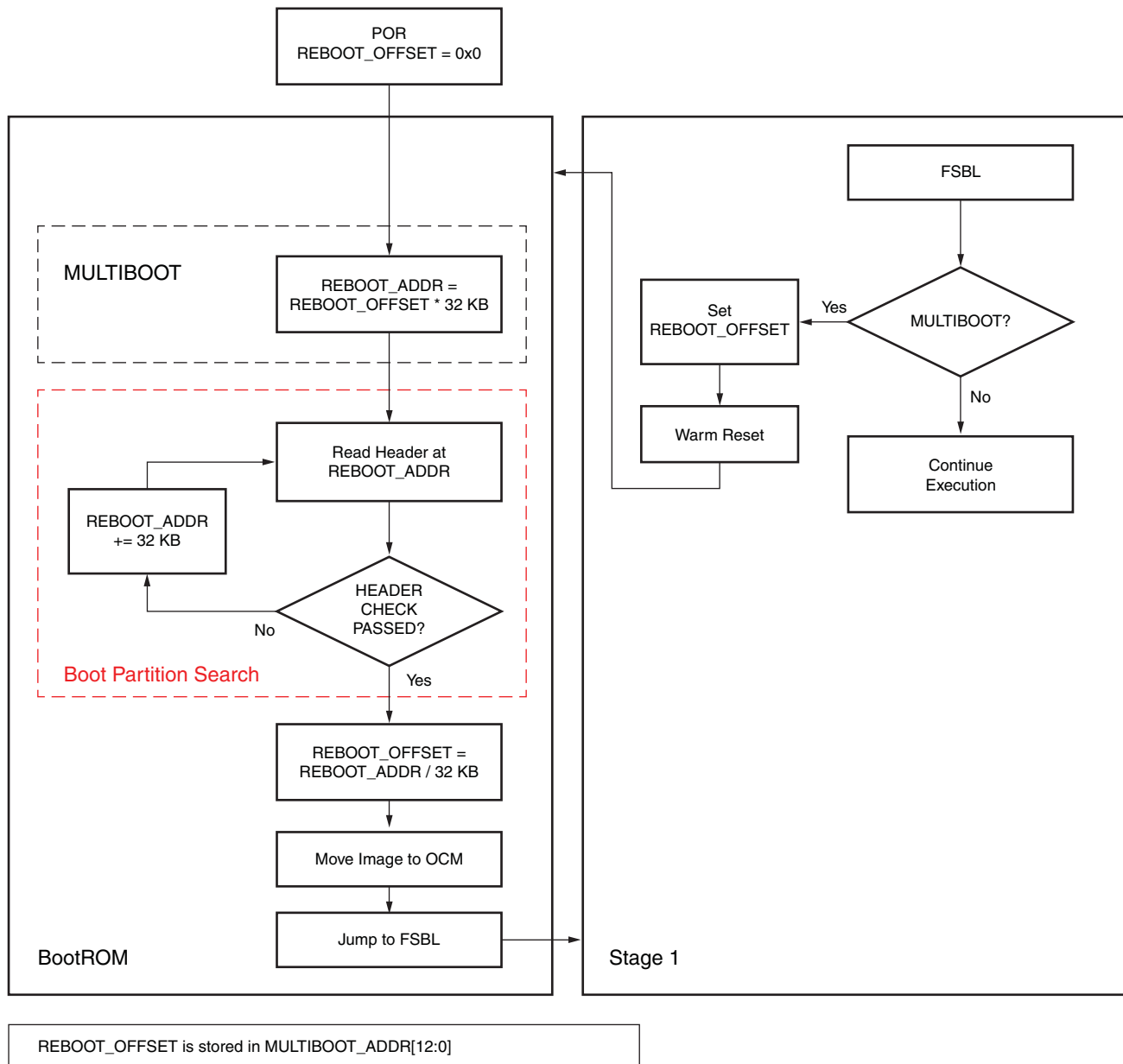
Table 6-8: SD MIO Pins

Signal Name	I/O	MIO Pin
SDIO0CLK	O	MIO[40]
SDIO0CMD	O	MIO[41]
SDIO0DATAIO[0:3]	IO	MIO[42:45]

For the BootROM to read BOOT.BIN the SD must be partitioned so that the first partition is a FAT file system. Additional non-FAT partitions are permitted, but the BootROM does not use the partitions to boot.

6.3.5 BootROM MultiBoot and Boot Partition Search

The BootROM supports partition search in the event of a failure in the primary BootROM partition. As part of the same mechanism for partition search, the BootROM supports multiboot where different images can be loaded on device reset. The multiboot and boot partition search follows the flow diagram in [Figure 6-2](#).



UG585_c6_02_071012

Figure 6-2: MultiBoot and Boot Partition Search

[Figure 6-2](#) demonstrates the multiboot flow from the stage 1 Image. The mechanism used in stage 1 could be used in later device stages as well.

To use multiboot multiple images must be placed in flash. Each image requires a BootROM header. These images can be placed in any order, but after POR the BootROM uses the first image in flash as the initial image.

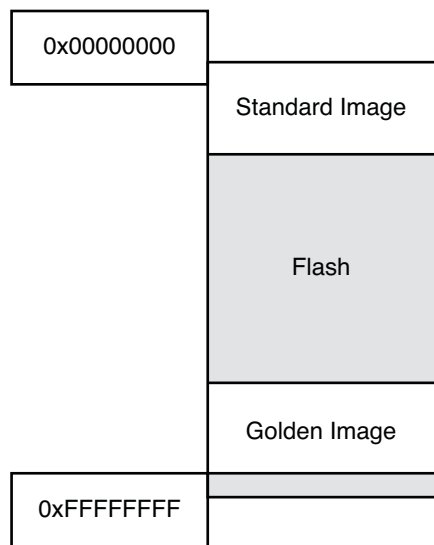
The steps to use multiboot are as follows:

1. Calculate the REBOOT_OFFSET address:
 - a. This offset is relative to the beginning of flash
 - b. REBOOT_OFFSET can be calculated as:

$$\text{Image Byte Address in Flash} / 0x8000$$

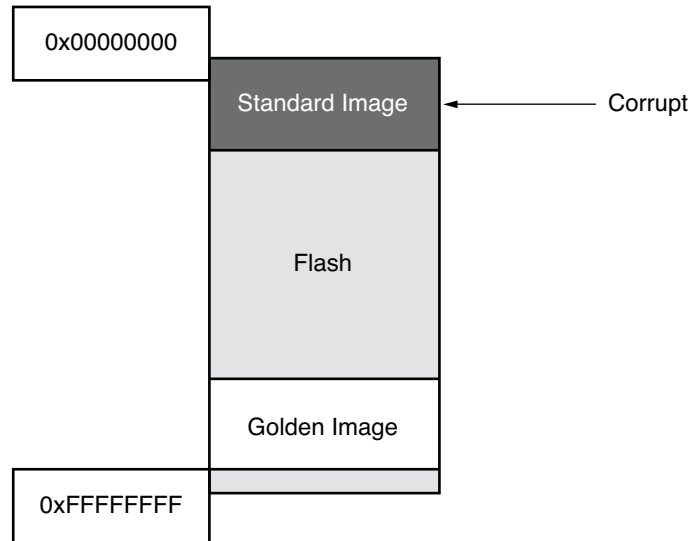
2. Write REBOOT_OFFSET to MULTIBOOT_ADDR[12:0].
3. Perform a warm reset.

For partition search to work, the golden images should be placed after the image intended for load in the absence of a failure. Figure 6-3 and Figure 6-4 demonstrate the flash layout required for boot partition search. Normally the device boots from the standard image but in the event of a failure the BootROM searches, finds, and boots from the golden image.



UG585_c6_03_031712

Figure 6-3: Standard Flash Layout



UG585_c6_04_031712

Figure 6-4: Corrupt Flash

The BootROM partition search mechanism protects against these failures:

- An update was started on the standard image but the system was interrupted after erasing the section requiring an update.
- The write operation began but the write process only partially finished.

The BootROM partition search mechanism does not protect against these failures:

- The section of memory where the standard image resides became corrupt by some unspecified means.
- A complete image was written but it is non-functional.

To provide a mechanism for fallback configuration, the BootROM provides the following functionality:

- The BootROM looks for a keyword in the header to identify a BootROM image ("XLNX" at offset 0x24 in header).
- The BootROM calculates a checksum to verify that the header is not corrupt. (Comparison value is located at 0x48 in header.)

If there is a failure in any of the checks, the BootROM begins searching on 32 KB boundaries for another image that passes the keyword test and the CRC check. The BootROM is only capable of handling one fallback image. Additionally, the BootROM only searches a limited address space on the configuration device:

- SD does not have golden image or search capabilities
- NAND searches the first 128 MB of memory
- NOR and QSPI search the first 16 MB of memory
- NOR and dual QSPI can use the first 32 MB of memory
- Single QSPI is limited to the first 16 MB of address space

6.3.6 Debug Status

The BootROM has two methods for communicating status information in the event of a startup failure. The primary method is through the REBOOT_STATUS register. Non-secure boot failures result in the BootROM disabling access to the AES engine, clearing the PL, and enabling JTAG.

After JTAG is enabled, the REBOOT_STATUS register can be read to find the source of the boot failure. The defined error output codes for the BootROM are identified in [Table 6-9](#).

Table 6-9: BootROM Error Output Codes

Name	Value	Description
BOOT_ROM_CRC_FAIL	0x1000	BOOT_ROM_CRC_FAIL occurs if the CRC unlock value created does not unlock the DevC block. This is a catastrophic error in that JTAG is not enabled and there is no recovery possible. This causes lockdown.
ILLEGAL_BOOT_MODE	0x8101	ILLEGAL_BOOT_MODE occurs if the mode selected is not recognized. This is possible if the value read from the BMR ends up changing in the processing of the BootROM. This causes lockdown.
PERIPHERAL_INIT_FAIL	0x8102	PERIPHERAL_INIT_FAIL occurs if the PCAP interface does not start up correctly and the BootROM times out. This causes lockdown.
ILLEGAL_RETURN	0x8103	ILLEGAL_RETURN occurs if the handoff functionality does not correctly start up the loaded application and the processor returns from the function when it is not supposed to. This causes lockdown.
DECRYPTION_FAILED	0x8110	DECRYPTION_FAILED and AUTHENTICATION_FAILED occur if there is a problem decrypting the secure boot image.
AUTHENTICATION_FAILED	0x8111	DECRYPTION_FAILED and AUTHENTICATION_FAILED occur if there is a problem decrypting the secure boot image.
IMAGE_HEADER_FORMAT_ERROR	0x8201	IMAGE_HEADER_FORMAT_ERROR occurs if there is a problem with the image that is not covered by the other error checks. This is also the status for a blank flash device. This causes lockdown.
REGISTER_INIT_FAILED	0x8202	REGISTER_INIT_FAILED occurs if there is an error in the data to be written to the registers by the BootROM prior to loading and starting the image. This error is triggered by a destination address being out of range. This causes lockdown.
BAD_IMAGE_START_ADDRESS	0x8203	BAD_IMAGE_START_ADDRESS occurs when the start address of the image is either not 0 or not in the proper address range for XIP. For example, having the start address indicate the linear Quad-SPI address range but the image is loaded into NOR. This causes lockdown.
EXECUTE_IN_PLACE_ERROR	0x8204	EXECUTE_IN_PLACE_ERROR occurs when either the image is encrypted or the boot source indicates NAND but the image header indicates that it is an XIP image. This causes lockdown.
BAD_IMAGE_CHECKSUM	0x8205	BAD_IMAGE_CHECKSUM occurs when the checksum of the header is not correct. This triggers the golden image hunt for NOR, NAND, and QSPI boot modes. This causes lockdown.
EXCEPTION_TAKEN_FAIL	0x8300	EXCEPTION_TAKEN_FAIL is catastrophic. This indicated that the BootROM had an exception triggered for some reason. Any interrupt or exception is considered an error. This causes lockdown.

6.3.7 Post BootROM State

The state of the PS after the BootROM executes depends on the following conditions:

- Whether secure mode is enabled
- What the mode pins were set to
- Whether there was a failure

The mode pins impact which MIO are enabled and what I/O standard they are set to after exiting the BootROM. Additionally, the mode setting impacts the boot peripheral settings. For example, if Quad-SPI is the selected boot source the needed MIO is enabled and the Quad-SPI controller is set with the necessary settings to read from flash. The modified values for each boot source are documented in the associated boot devices section.

The general processor state upon BootROM exit is as follows:

- MMU, Icache, Dcache, L2 cache are all disabled
- Both processors are in the supervisor state
- ROM code is masked and inaccessible
- 192 KB of OCM is accessible starting at address 0x0 while 64 KB is accessible starting at address 0xFFFF0000
- CPU0 branches into the stage 1 image if no failure takes place
- CPU1 is in a WFE state while executing code located at address 0xFFFFFE00 to 0xFFFFFFF0

The full list of registers modified by the BootROM for JTAG boot mode is listed in [Table 6-10](#). Boot modes that use other configuration interfaces have MIO registers and peripheral registers modified for the associated configuration interface.

Table 6-10: BootROM Modified Registers

Address	Post BootROM Value	Register Name
0xf8007000	0x4e00e07f	devcfg.CTRL
0xf8007004	0x0000001a	devcfg.LOCK
0xf8007008	0x00000508	devcfg.CFG
0xf800700c	0xf8020006	devcfg.INT_STS
0xf8007014	0x40000f30	devcfg.STATUS
0xf8007028	0xffffffff	devcfg.ROM_SHADOW
0xf8007034	0x757bdf0d	devcfg.UNLOCK
0xf8007080	0x10800000	devcfg.MCTRL
0xf8f02104	0x02060000	l2cache.reg1_aux_control
0xf8f02f40	0x00000004	l2cache.reg15_debug_ctrl
0xf8f00040	0x00100000	mpcore.Filtering_Start_Address_Register
0xf8f00044	0xffe00000	mpcore.Filtering_End_Address_Register
0xf8f00108	0x00000002	mpcore.ICCBPR

Table 6-10: BootROM Modified Registers (Cont'd)

Address	Post BootROM Value	Register Name
0xf8f00200	0x9dfec6f5	mpcore.Global_Timer_Counter_Register0
0xf8f00204	0x00000002	mpcore.Global_Timer_Counter_Register1
0xf8f00208	0x00000001	mpcore.Global_Timer_Control_Register
0xf8000258	0x00400002	slcr.REBOOT_STATUS
0xf8000530	0x03731093	slcr.PSS_IDCODE
0xf8000910	0x00000018	slcr.OCM_CFG
0xf8000a1c	0x00010101	slcr.L2C_RAM
0xf8000a90	0x01010101	slcr.OCM_RAM
0xf8000b70	0x00000020	slcr.DDRIOB_DCI_CTRL
0xe0001000	0x00000114	uart1.Control_reg0
0xe0001004	0x00000020	uart1.mode_reg0
0xe0001014	0x00000208	uart1.Chnl_int_sts_reg0
0xe0001018	0x00000056	uart1.Baud_rate_gen_reg0
0xe0001028	0x000000fb	uart1.Modem_sts_reg0
0xe000102c	0x0000000a	uart1.Channel_sts_reg0
0xe0001034	0x00000004	uart1.Baud_rate_divider_reg0

Post BootROM Security

If secure mode is enabled the AES engine is accessible post BootROM. Conversely if the Zynq-7000 device boots non-secure the AES engine is disabled. By default, upon completion of a secure PS first stage boot, the encryption logic remains enabled. It is the user's responsibility to safeguard the PCAP/ICAP interfaces while AES is enabled.

The BootROM locks a number of bits in the DevC module prior to exiting to ensure security. The bits the BootROM locks are listed in [Table 6-11](#).

Table 6-11: DEV_CFG Lock Register

Bit Position	Bit Name	Lock Status
4	AES_FUSE_LOCK	1
3	AES_EN_LOCK	1 ⁽¹⁾
2	SEU_LOCK	0
1	SEC_LOCK	1
0	DBG_LOCK	0

Notes:

1. The AES_EN_LOCK bit is only locked in NON_SECURE boot.

Post BOOTROM Debug

In the event of a failure while booting non-secure the BootROM enables JTAG access so that the RE-BOOT_STATUS register can be read. A debugging tool like XMD will have full access to the processor when JTAG is enabled.

If a failure occurs while booting in secure mode the BootROM disables the AES engine, clears the OCM, clears the PL, and halts the processor. JTAG is not enabled, consequently, the REBOOT_STATUS value is not available to be read.

Starting Code on the CPU1

CPU0 is in charge of starting any additional code on CPU1. The BootROM keeps CPU1 essentially in the reset state from power-on-reset. Nothing has been enabled and only a few general purpose registers have been modified to place it in a state where it is waiting at the WFE instruction.

There is a small amount of protocol required for CPU0 to start an application on CPU1. When CPU1 receives a system event, it immediately reads the contents of address `0xFFFFFFFF0` and jumps to that address. If the SEV is issued prior to updating the destination address location (`0xFFFFFFFF0`), CPU1 continues in the WFE state because `0xFFFFFFFF0` has the address of the WFE instruction as a safety net. If the data that is written to address `0xFFFFFFFF0` is invalid or points to uninitialized memory, results are unpredictable.

Only ARM-32 ISA code is supported for the initial jump on CPU1. Thumb and Thumb-II code is not supported at the destination of the jump. This means that the destination address must be 32-bit aligned and must be a valid ARM-32 instruction. If these conditions are not met, results are unpredictable.

The steps for CPU0 to start an application on CPU1 are as follows:

1. Write the address of the application for CPU1 to `0xFFFFFFFF0`
2. Execute the SEV instruction to cause CPU1 to wake up and jump to the application.

The address range `0xFFFFFEE00` to `0xFFFFFFFF0` is reserved and not available for use until the stage 1 or above application is fully functional. Any access to these regions prior to the successful startup of the second CPU causes unpredictable results.

6.4 Device Configuration Interface

The device configuration interface (DevC), shown in [Figure 6-5](#), consists of three main blocks that operate independently:

- An AXI-PCAP bridge for interfacing to the PL configuration logic
- Device security management
- An XADC interface

The device configuration interface also contains an APB interface used by the host to configure the three blocks, to access the overall status, and to communicate with the PL XADC.

6.4.1 Block Diagram

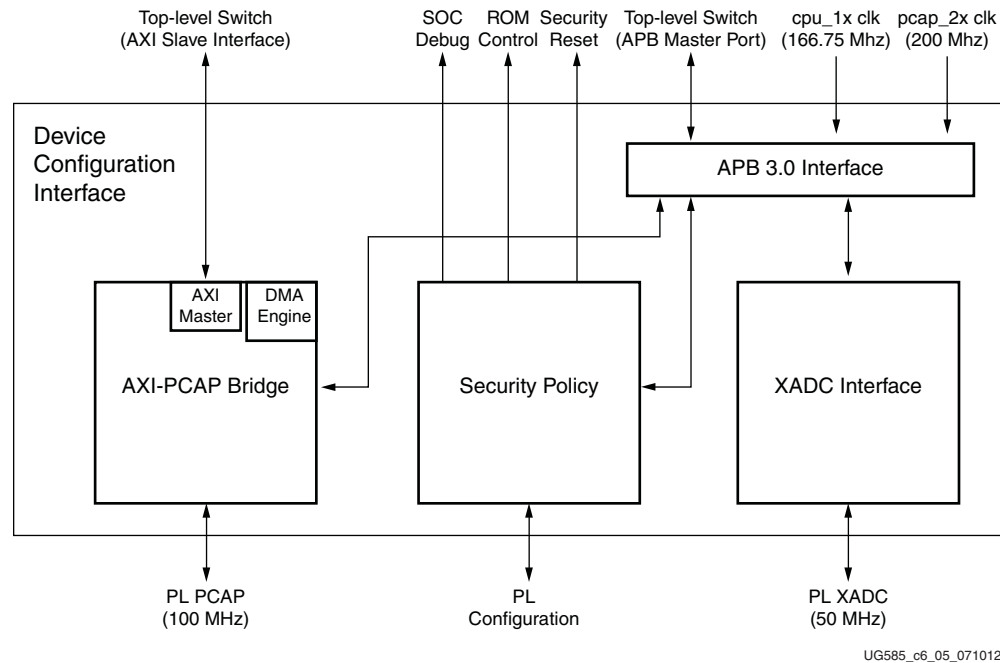


Figure 6-5: DevC Block Diagram

6.4.2 Features

The device configuration interface manages basic device security and provides a simple DMA interface, PS setup, and PL configuration. The DevC:

- Enables PL configuration through the processor configuration access port (PCAP) in both secure and non-secure master boot
- Supports PL configuration readback
- Supports concurrent bitstream download/upload
- Supports 400 MB/s PCAP download throughput for non-secure PL configuration and 100 MB/s for secure PL configuration
- Enforces Zynq-7000 device system-level security including debug security
- Supports XADC serial interface
- Supports XADC alarm and over-temperature interrupt
- Supports 200 MHz AXI word-aligned burst read and write transfers
- Secure boot ROM code protection

6.4.3 Functional Description

AXI-PCAP Bridge

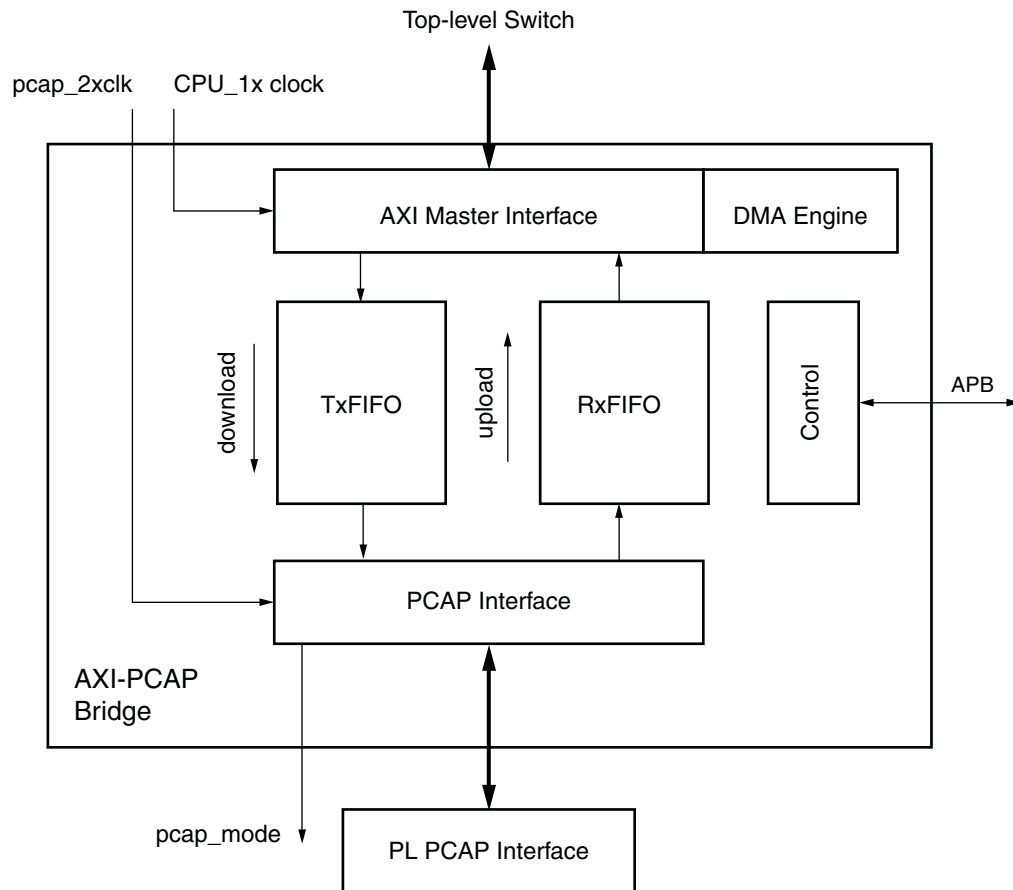
The AXI-PCAP bridge converts 32-bit AXI formatted data to the 32-bit PCAP protocol and vice versa. The bridge supports both concurrent and non-concurrent download and upload of configuration data. A transmit and receive FIFO buffer data between the AXI and PCAP interfaces. A DMA engine moves data between the FIFOs and a memory device, typically the on-chip RAM, the DDR memory, or one of the peripheral memories. Non-secure data to the PCAP interface can be sent every clock cycle, encrypted data can be sent every four clock cycles.

To transfer data across the PCAP interface, the PL must be powered on. The PCAP interface is enabled using the PCAP_MODE and PCAP_PR bits of the DevC's Control register. If encrypted data is being sent, the QUARTER_PCAP_RATE_EN bit should also be set.

Data is transferred through the PCAP interface using the DevC's built in DMA engine. To start a transfer, the four DMA registers must be written in this order:

1. DMA Source Address register
2. DMA Destination Address register
3. DMA Source Length register
4. DMA Destination Length register

To transfer data to the PL via the PCAP interface, the destination address should be `0xFFFF_FFFF`. Similarly, to read data from the PL via the PCAP interface, the source address should be `0xFFFF_FFFF`. Encrypted PS images must also be sent across the PCAP interface because the AES and HMAC engines reside within the PL. In this case, the DMA source address should be an external memory interface and the destination address should be on-chip RAM.



UG585_c6_06_072412

Figure 6-6: AXI-PCAP Bridge

The DevC's DMA engine can also be used to load non-secure PS images. Before doing this, the INT_PCAP_LPBK bit should be set in the Miscellaneous Control register. This bit enables an internal loopback that bypasses the PCAP interface. This bit needs to be disabled again before using the PCAP interface. The DMA source address should be an external memory and the destination address should be on-chip RAM or a valid external interface like DDR.

The PCAP interface can also be used to perform a PL configuration readback. To perform a readback, the PS must be running software capable of generating the correct PL readback commands. Two DMA accesses are required to complete a PL configuration readback. The first access is used to issue the readback command to the PL. The second access is needed to read the PL readback data from the PCAP. The smallest amount of data that can be read back from the PL is one configuration frame which contains 101 32-bit words.

Sample first DMA access for PL readback:

1. DMA Source Address – location of PL readback command sequence
2. DMA Destination Address – desired location to store readback data, note that the on-chip RAM is not large enough to hold a complete PL readback
3. DMA Source Length – number of commands in the PL readback command sequence
4. DMA Destination Length – number of readback words expected from the PL

There are three limitations when accessing the PL configuration systems:

1. Readback of configuration registers or data cannot be performed until the PCFG_DONE is received (bit-2 of the INT_STS register).
2. A single PCAP access cannot be split across multiple DMA accesses. If the readback command sent to the PL requests 505 words, the DevC's DMA must also be setup to process 505 words. Splitting the transaction into two DMA accesses results in data loss and unexpected DMA behavior.
3. The DevC DMA must have sufficient bandwidth to process the PL readback data due to a lack of data flow control on the PL side of the PCAP. Overflow of the PCAP RxFIFO results in data loss and unrecoverable DMA behavior. If adequate bandwidth cannot be allocated to the DevC DMA, then the PCAP clock could be slowed down or the readback could be broken up into multiple smaller transactions.

For more information regarding PL configuration readback, see [UG470](#), *7 Series FPGAs Configuration User Guide*.

Device Security Management

The DevC contains a security policy block that provides these functions:

- Monitors the system security and can assert a security reset when conflicting status is detected that could indicate inconsistent system configuration or tampering
- Controls and monitors the PL configuration logic via the APB interface
- Controls the ARM CoreSight's debug access port (DAP) and debug levels
- Provides on-chip ROM shadow control

See [Chapter 32, Device Secure Boot](#) for more information.

XADC Interface

The XADC interface provides these functions and features:

- Reads and writes XADC registers
- 15-deep write command FIFO and 15-deep read FIFO (both 32-bits wide)
- Programmable FIFO-level interrupt
- Alarm interrupt
- Over-temperature interrupt

See [Chapter 30, Analog-to-Digital Converter Interface](#) for more information.

6.4.4 Device Configuration Flow

PS Master Non-Secure Boot

In this boot mode the PS acts as the device master and the boot ROM loads a plain text PS image from the selected external memory (see Figure 6-7). The PL is not required to be powered on and a PL bitstream can be loaded with the PS image right away or at some later time.

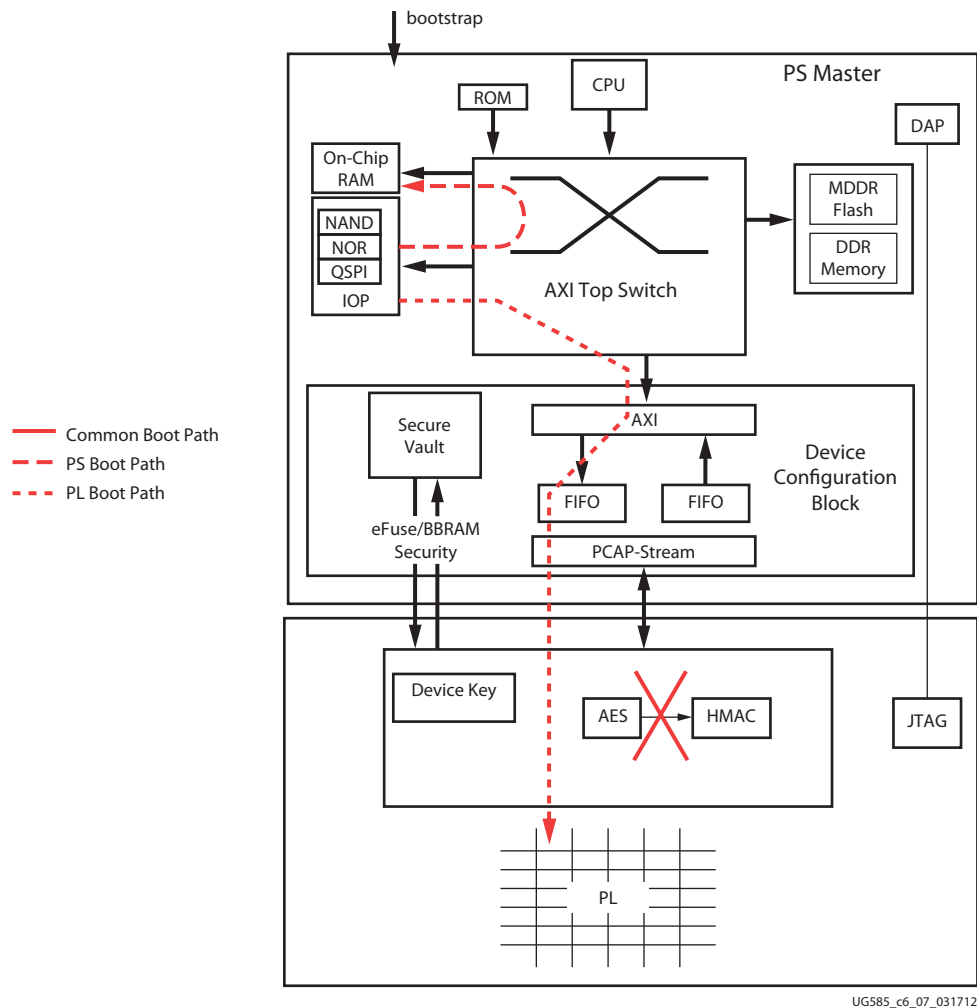


Figure 6-7: PS Master Non-Secure Boot Diagram

The configuration flow is:

1. Device power-on reset
2. Boot ROM execution
 - a. Reads bootstrap to determine external memory interface
 - b. Reads boot header to determine encryption status and image destination
3. Boot ROM uses the DevC's DMA to load the first stage boot loader (FSBL) into on-chip RAM or other valid destination

4. Boot ROM shuts down, releases CPU control to the FSBL
5. (Optional) FSBL loads PL bitstream via PCAP

PS Master Secure Boot

In this boot mode the PS acts as the device master and the boot ROM loads an encrypted PS image from the selected external memory (see Figure 6-8). Because the AES and HMAC engines reside in the PL, the PL is required to be powered on for the initial boot sequence. The Boot ROM verifies that the PL has power before attempting to decrypt the FSBL. After the PS has booted, the PL can be configured using an encrypted bitstream or it can be powered down to be configured later.

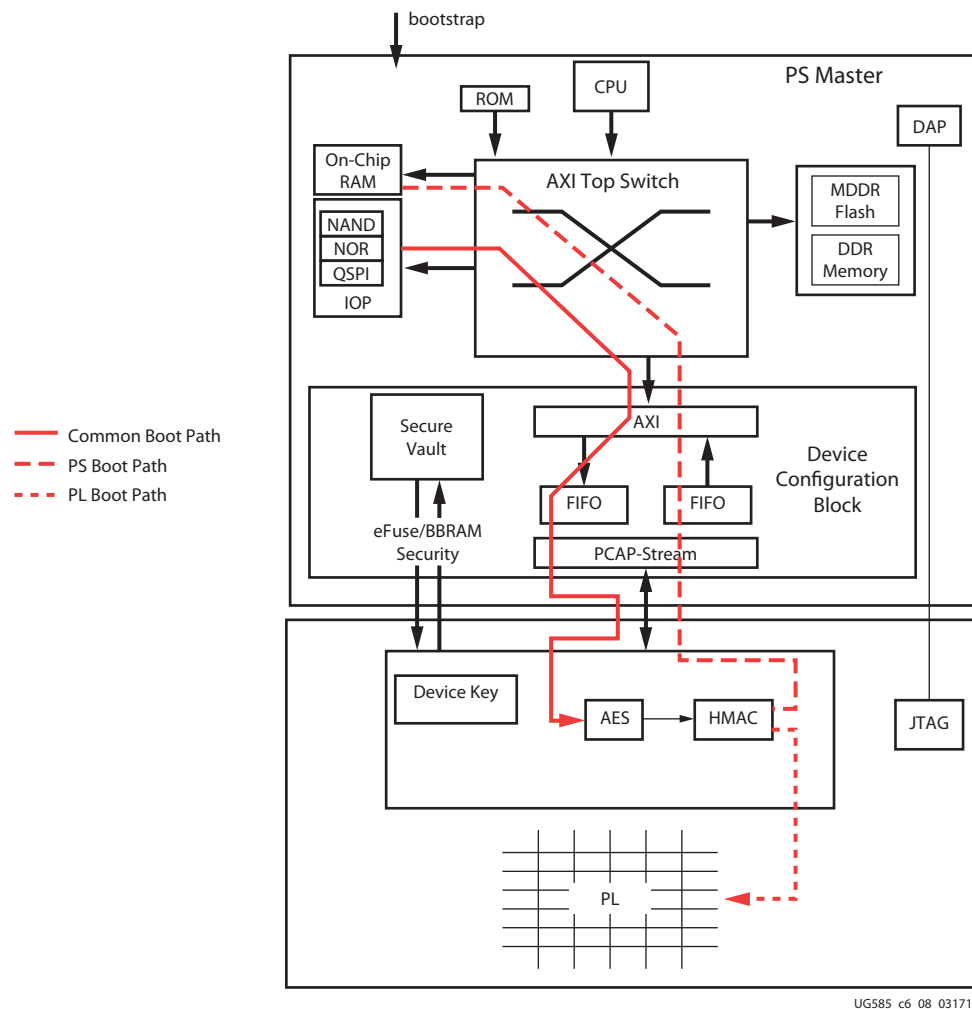


Figure 6-8: PS Master Secure Boot Diagram

The configuration flow is:

1. Device power-on reset, including power-on of PL
2. Boot ROM execution
 - a. Reads bootstrap to determine external memory interface

- b. Reads boot header to determine encryption status (secure)
 - c. Ensures PL is powered on to begin FSBL decryption
3. Boot ROM uses the DevC's DMA to send the encrypted FSBL to the AES and HMAC in the PL via PCAP
4. The PL uses the PCAP to return the decrypted FSBL to the PS where it is loaded to the on-chip RAM
5. Boot ROM shuts down, releases CPU control to the FSBL
6. (Optional) FSBL configures the PL with an encrypted bitstream

PS JTAG Non-Secure Boot

In this boot mode the PS is a slave to the JTAG port (see [Figure 6-9](#)). The JTAG mode must be set to cascade, booting from split mode is not possible. The PL's external JTAG pins are used requiring that the PL be powered up. The AES engine is disabled, secure images are not allowed. The PS can be booted independently or the PS and PL can be configured at the same time. If the PS is booted independently, the PL TAP controller instruction and data registers must be accounted for when data is shifted between TDI and TDO. In the cascaded configuration, the PS DAP registers are the last to be shifted in. After the PS has booted, the PL can be configured or powered down to be configured later.

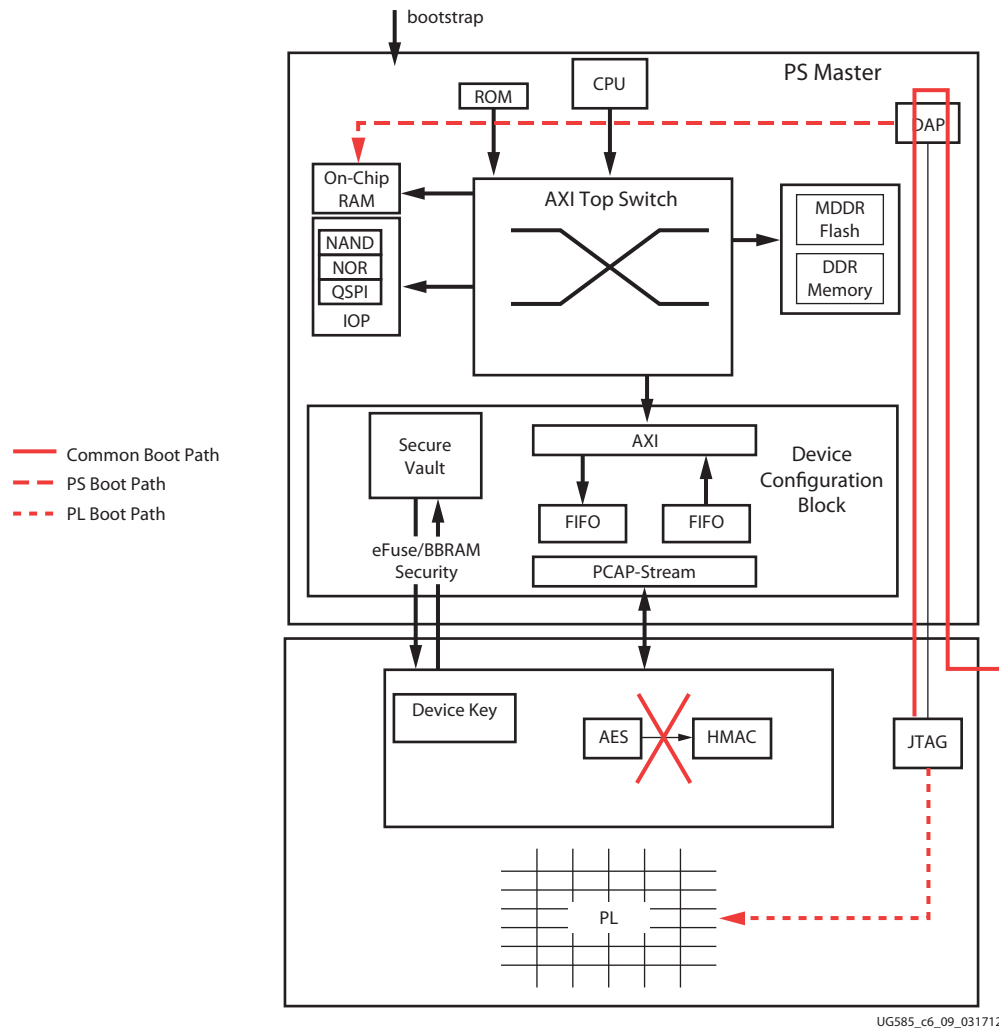


Figure 6-9: PS JTAG Non-Secure Boot Diagram

The configuration flow is:

1. Device power-on reset, including power-on of PL
2. Boot ROM execution
 - a. Reads bootstrap to determine boot mode, JTAG mode must be cascaded
 - b. Performs CRC self-check
3. Disables all security features, enables DAP controller and JTAG chain
4. Boot ROM shuts down, releases CPU control to JTAG
5. JTAG port is used to load PS image
6. (Optional) JTAG or PS image is used to configure PL

6.4.5 PL Configuration

With the exception of JTAG, the PL must always be configured using the PCAP. Users are free to configure the PL at any time, whether it is at PS boot or at some later time using an image loaded into the PS. This section describes the requirements for configuring the PL.

Determining the PL Power Status

The PL must first be powered on before it can be configured. When power is applied to the PL it begins its independent power-on reset sequence followed by its housecleaning function which clears all of the PL's configuration SRAM cells. The power-on reset status of the PL can be monitored by the PS.

The power status of the PL is tracked in the DevC's Miscellaneous Control register, bit 8. If this bit is set, then the PL has power. The PL power status can also be tracked as an interrupt. Bit 29 of the DevC's Control register, PCFG_POR_CNT_4K, can be used to speed up the PL's power on reset stage.

Additional information about the PL power-up status can be obtained by reading bit 5 of the DEVCI Status register. If the bit is Low, then the PL is in a reset state. A transition from Low to High indicates the start of the housecleaning function.

At any time, the PCFG_PROG_B bit on the Control register can be used to issue a global reset to the PL. If this bit is set Low, the PL begins its housecleaning sequence, PCFG_INIT is held Low until the PCFG_PROG_B bit is set High.

Configuring the PL via PCAP

To send a configuration bitstream, the PL must be finished with its housecleaning function. This status is indicated by the PCFG_INIT bit in the DevC's Status register. The internal loopback function must also be disabled and the PCAP_MODE must be set. The bitstream is transferred to the PL using the DevC DMA. A successful PL configuration is indicated by the PCFG_DONE flag on the Interrupt Status register. This flag also generates a system interrupt.

The configuration flow is:

1. Wait for PCFG_INIT to be set High by the PL (STATUS-bit[4])

2. Set internal loopback to 0 (MCTRL-bit[4])
3. Set PCAP_PR and PCAP_MODE to 1 (CTRL-bit[27 and 26])
4. Initiate a DevC DMA transfer
 - a. Source address: Location of PL bitstream
 - b. Destination address: 0xFFFF_FFFF
 - c. Source length: Total number of 32-bit words in the PL bitstream
 - d. Destination length: Total number of 32-bit words in the PL bitstream
5. Wait for PCFG_DONE to be set High by the PL (INT_STS-bit[2])

PL Reconfiguration

After the PL has been configured, it can be reconfigured using either the PCAP or ICAP. To use the ICAP, the PCAP_PR, bit 27 of the Control register, must be set Low. The initial configuration of the PL must also have connected the ICAP to an external interface or to some logic internal to the PL.

To reconfigure the PL using PCAP, the PCAP_MODE and PCAP_PR bits must be set to 1 in the Control register. The internal loopback function must also be disabled. The reconfiguration bitstream is sent to the PL using the DevC DMA, the source and destination is the same as for the initial configuration.

PL reconfiguration flow using PCAP:

1. Disable the AXI interface to the PL.
2. Disable the PL level shifters by writing 0xA to the SLCR LVL_SHFTR_EN register.
3. Set PCAP_MODE and PCAP_PR High.
4. Clear the previous configuration from the PL (optional):
 - a. Set PCFG_PROG_B High.
 - b. Set PCFG_PROG_B Low.
 - c. Check for PCFG_INIT = 0 (STATUS-bit[4]).
 - d. Set PCFG_PROG_B High.
5. Check for PCFG_INIT = 1 (STATUS-bit[4]).
6. Set INT_PCAP_LPBK Low (MCTRL-bit[4]).
7. Initiate a DevC DMA transfer:
 - a. Source Address: Location of new PL bitstream.
 - b. Destination Address: 0xFFFF_FFFF.
 - c. Source Length: Total number of 32-bit words in the new PL bitstream.
 - d. Destination Length: Total number of 32-bit words in the new PL bitstream.
8. Clear PCFG_DONE_INT by writing a 1 to the INT_STS[2].
9. Wait for PCFG_DONE_INT to be set High.
10. Enable the PL level shifters by writing 0xF to the SLCR LVL_SHFTR_EN register.
11. Enable the AXI interface to the PL.

The PCFG_DONE flag must be cleared from the Interrupt Status register after starting the PCAP or ICAP reconfiguration so that the successful reconfiguration of the PL can be detected by the PS. It is important to note that the PCAP and ICAP interfaces are mutually exclusive. Only one interface can talk to the PL configuration controller at a time. Which interface is in use is controlled by the PCAP_PR bit in the DevC's Control register. It is possible to use both interfaces in the device, care should be taken to ensure that all outstanding transactions have completed before changing interfaces.

Configuring the PL via JTAG

The PL can be configured using JTAG in either cascade or split mode. To use either JTAG mode, the JTAG chain disable bit on the Control register must be set to 0. In split mode, the PL behaves like a Xilinx 7 Series FPGA. In cascade mode, the PS DAP controller must also be enabled using the DAP enable bits on the Control register. The PS DAP instruction and data registers must also be accounted for. Additional information regarding PL configuration via JTAG can be found in [UG470, 7 Series FPGAs Configuration User Guide](#).

PL Secure Configuration

To perform a secure configuration of the PL, the PS must have booted securely. The AES and HMAC engines can only be enabled by the Boot ROM. The procedure for loading a secure bitstream is the same as any other bitstream but the quarter PCAP rate enable bit must be set in the Control register. Because the AES engine only de-crypts one byte at a time, the PCAP can only send one 32-bit word to the PL every four clock cycles.

6.4.6 Register Overview

[Table 6-12](#) provides an overview of the device configuration registers.

Table 6-12: Device Configuration and Boot Register Overview

Function	Description	Hardware Register	Offset	Type
Device Config	Control	CTRL	0x000	R/W
	Sticky Locks require POR to reset	LOCK	0x004	R/Sticky Write
	Configuration	CFG	0x008	R/W
Device Config DMA	Interrupt Status	INT_STS	0x00C	R + Clr or Wr
	Interrupt Mask	INT_MASK	0x010	R/W
	Status	STATUS	0x014	R + Clr or Wr
	DMA Source Address	DMA_SRC_ADDR	0x018	R/W
	DMA Destination Address	DMA_DEST_ADDR	0x01C	R/W
	DMA Source Length	DMA_SRC_LEN	0x020	R/W
	DMA Destination Length	DMA_DEST_LEN	0x024	R/W
Boot	Multi-Boot Offset	MULTIBOOT_ADDR	0x02C	R/W
	Software ID Register	SW_ID	0x030	R/W
	Miscellaneous Control	MCTRL	0x080	R/W

Interrupts

7.1 Environment

This chapter describes the system-level interrupt environment and the functions of the interrupt controller (see [Figure 7-1](#)). The PS is based on ARM architecture, utilizing two Cortex-A9 processors (CPUs) and the GIC pl390 interrupt controller. The interrupt structure is closely associated with the CPUs and accepts interrupts from the I/O peripherals (IOP) and the programmable logic (PL). This chapter includes these key topics:

- Private, shared and software interrupts
- GIC functionality
- Interrupt prioritization and handling

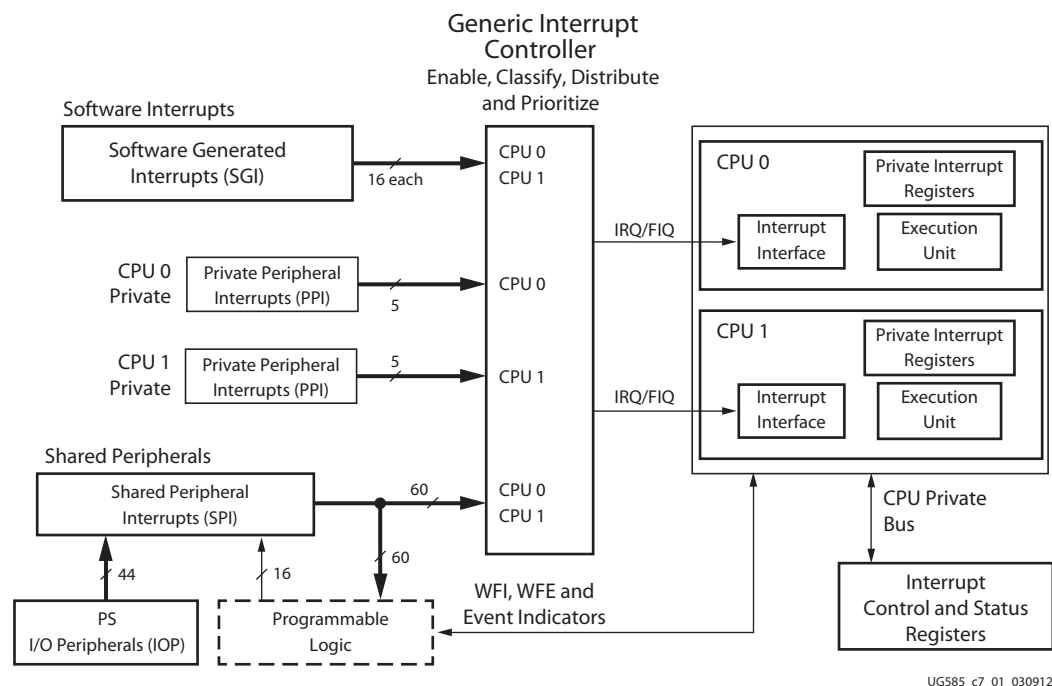


Figure 7-1: System-Level Block Diagram

7.1.1 Private, Shared and Software Interrupts

Each CPU has a set of private peripheral interrupts (PPIs) with private access using banked registers. The PPIs include the global timer, private watchdog timer, private timer, and FIQ/IRQ from the PL. Software generated interrupts (SGIs) are routed to one or both CPUs. The SGIs are generated by writing to the ICDSGIR register. The shared peripheral interrupts (SPIs) are generated by the various I/O and memory controllers in the PS and PL. They are routed to either or both CPUs. The SPI interrupts from the PS peripherals are also routed to the PL.

7.1.2 Generic Interrupt Controller (GIC)

The generic interrupt controller (GIC) is a centralized resource for managing interrupts sent to the CPUs from the PS and PL. The controller enables, disables, masks, and prioritizes the interrupt sources and sends them to the selected CPU (or CPUs) in a programmed manner as the CPU interface accepts the next interrupt. In addition, the controller supports security extension for implementing a security-aware system.

The controller is based on the ARM Generic Interrupt Controller Architecture version 1.0 (GIC v1), non-vectored.

The registers are accessed via the CPU private bus for fast read/write response by avoiding temporary blockage or other bottlenecks in the interconnect.

The interrupt distributor centralizes all interrupt sources before dispatching the one with the highest priority to the individual CPUs. Hardware ensures that an interrupt targeted to several CPUs can only be taken by one CPU at a time. All interrupt sources are identified by a unique interrupt ID number. All interrupt sources have their own configurable priority and list of targeted CPUs.

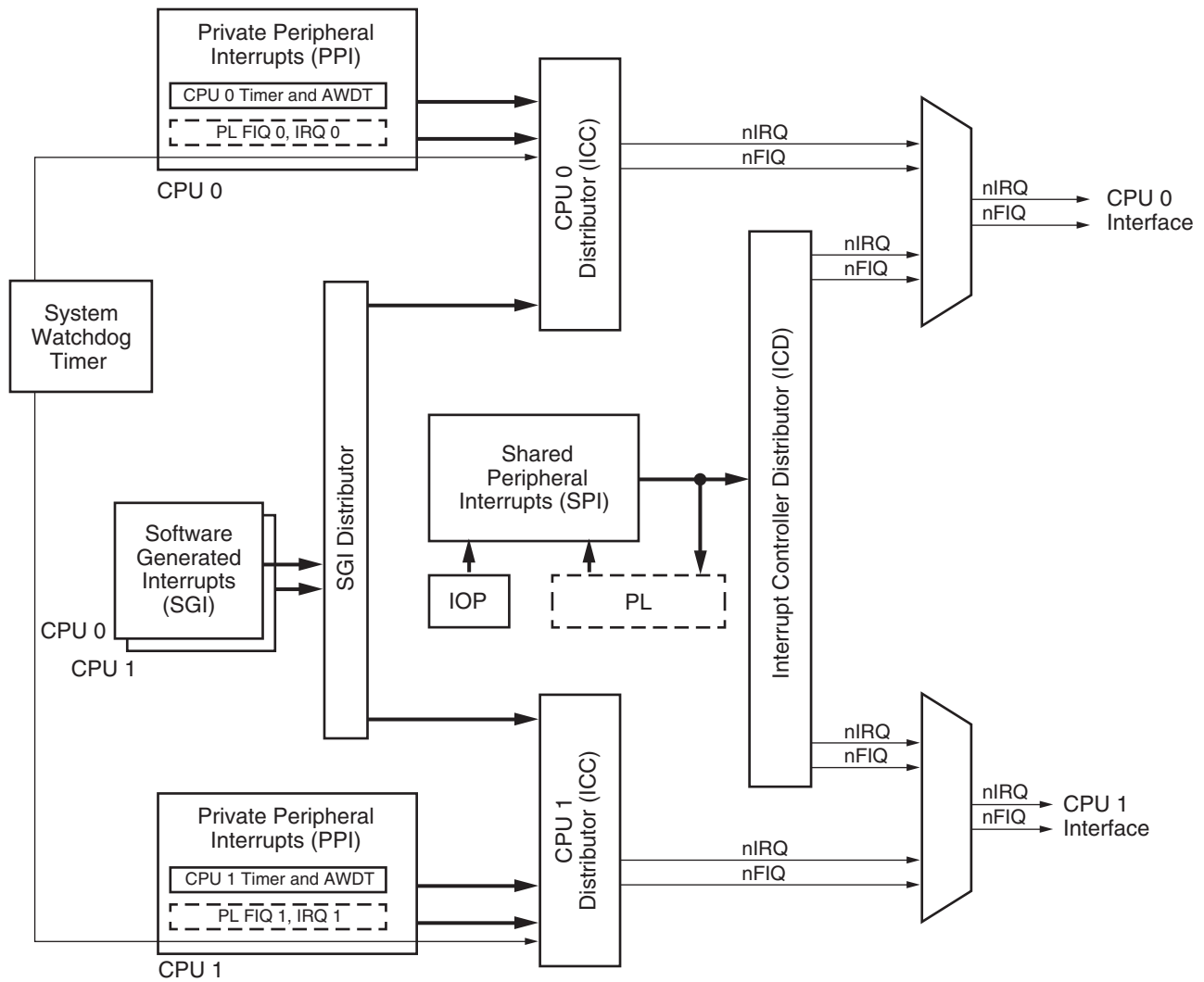
7.1.3 Resets and Clocks

The interrupt controller is reset by the reset subsystem by writing to the PERI_RST bit of the A9_CPU_RST_CTRL register in the SLCR. The same reset signal also resets the CPU private timers and private watchdog timers (AWDT).

The interrupt controller operates with the CPU_3x2x clock (half the CPU frequency).

7.1.4 Block Diagram

The shared peripheral interrupts are generated from various system subsystems that include the I/O peripherals in the PS and logic in the PL. The interrupt sources are illustrated in [Figure 7-2](#).



UG585_c7_02_030912

Figure 7-2: Interrupt Controller Block Diagram

7.2 Functional Description

7.2.1 Software Generated Interrupts (SGI)

Each CPU can interrupt itself, the other CPU, or both CPUs using a software generated interrupt (SGI). There are 16 software generated interrupts (see [Table 7-1](#)). An SGI interrupt is generated by writing the SGI interrupt number to the ICDSGIR register and specifying the target CPU(s). This write occurs via the CPU's own private bus. Each CPU has its own set of SGI registers to generate one or more of the 16 software generated interrupts. The CPU can interrupt itself, the other CPU, or both CPUs. The interrupts are cleared by reading the ICCIAR (Interrupt Acknowledge) register or writing a 1 to the corresponding bits of the ICDICPR (Interrupt Clear-Pending) register.

All SGIs are edge triggered. The sensitivity types for SGIs are fixed and cannot be changed; the ICDICFR0 register is read-only.

Table 7-1: Software Generated Interrupts (SGI)

IRQ ID#	Name	SGI#	Type	Description
0	Software 0	0	Rising edge	A set of 16 interrupt sources that are private to each CPU that can be routed to up to 16 common interrupt destinations where each destination can be one or more CPUs.
1	Software 1	1	Rising edge	
~	...	~	...	
15	Software 15	15	Rising edge	

7.2.2 CPU Private Peripheral Interrupts (PPI)

Each CPU connects to a private set of five shared peripheral interrupts. The PPIs are listed in [Table 7-2](#).

All interrupt sensitivity types are fixed and cannot be changed. Note that the fast interrupt (FIQ) signal and the interrupt (IRQ) signal from the PL are inverted and then sent to the interrupt controller. Therefore, they are active High at the PS-PL interface, although the ICDICFR1 register reflects them as active Low sensitive.

Table 7-2: Private Peripheral Interrupts (PPI)

IRQ ID#	Name	PPI#	Type	Description
26:16	Reserved	~	~	Reserved
27	Global Timer	0	Rising edge	Global timer
28	nFIQ	1	Active Low level (active High at PS-PL interface)	Fast interrupt signal from the PL: CPU0: IRQF2P[18] CPU1: IRQF2P[19]
29	CPU Private Timer	2	Rising edge	Interrupt from private CPU timer

Table 7-2: Private Peripheral Interrupts (PPI) (Cont'd)

IRQ ID#	Name	PPI#	Type	Description
30	AWDT{0, 1}	3	Rising edge	Private watchdog timer for each CPU
31	nIRQ	4	Active Low level (active High at PS-PL interface)	Interrupt signal from the PL: IRQF2P[16] CPU1: IRQF2P[17]

7.2.3 Shared Peripheral Interrupts (SPI)

A group of approximately 60 interrupts from various modules can be routed to one or both of the CPUs or the PL. The interrupt controller manages the prioritization and reception of these interrupts for the CPUs.

The reset default value for all of the SPI interrupt types is an active High level. However, software is required to program interrupts 32, 33 and 92 to rising-edge sensitivity using the ICDICFR2 and ICDICFR5 registers. The SPI interrupts are listed in Table 7-3.

Table 7-3: PS and PL Shared Peripheral Interrupts (SPI)

Source	Interrupt Name	IRQ ID#		Type	PS-PL Signal Name	I/O
		Decimal	Hex			
APU	CPU 1, 0 (L2, TLB, BTAC)	33:32	21h:20h	Rising edge	~	~
	L2 Cache	34	22h	High level	~	~
	OCM	35	23h	High level	~	~
Reserved	~	36	24h	~	~	~
PMU	PMU [1,0]	38, 37	25h:26h	High level	~	~
XADC	XADC	39	27h	High level	~	~
DVI	DVI	40	28h	High level	~	~
SWDT	SWDT	41	29h	High level	~	~
Timer	TTC 0	43:42	2B:2Ah	High level	~	~
Reserved	~	44	2Ch	~	~	~
DMAC	DMAC Abort	45	2Dh	High level	IRQP2F[28]	Output
	DMAC [3:0]	49:46	31h:2Eh	High level	IRQP2F[23:20]	Output
Memory	SMC	50	32h	High level	IRQP2F[19]	Output
	Quad SPI	51	33h	High level	IRQP2F[18]	Output
Debug	CTI	~	~	High level	IRQP2F[17]	Output

Table 7-3: PS and PL Shared Peripheral Interrupts (SPI) (Cont'd)

Source	Interrupt Name	IRQ ID#		Type	PS-PL Signal Name	I/O
		Decimal	Hex			
IOP	GPIO	52	34h	High level	IRQP2F[16]	Output
	USB 0	53	35h	High level	IRQP2F[15]	Output
	Ethernet 0	54	36h	High level	IRQP2F[14]	Output
	Ethernet 0 Wakeup	55	37h	High level	IRQP2F[13]	Output
	SDIO 0	56	38h	High level	IRQP2F[12]	Output
	I2C 0	57	39h	High level	IRQP2F[11]	Output
	SPI 0	58	3Ah	High level	IRQP2F[10]	Output
	UART 0	59	3Bh	High level	IRQP2F[9]	Output
	CAN 0	60	3Ch	High level	IRQP2F[8]	Output
PL	FPGA [7:0]	68:61	44h:3Dh	High level	IRQF2P[7:0]	Input
Timer	TTC 1	71:69	47h:45h	High level	~	~
DMAC	DMAC[7:4]	75:72	4Bh:48h	High level	IRQP2F[27:24]	Output
IOP	USB 1	76	4Ch	High level	IRQP2F[7]	Output
	Ethernet 1	77	4Dh	High level	IRQP2F[6]	Output
	Ethernet 1 Wakeup	78	4Eh	High level	IRQP2F[5]	Output
	SDIO 1	79	4Fh	High level	IRQP2F[4]	Output
	I2C 1	80	50h	High level	IRQP2F[3]	Output
	SPI 1	81	51h	High level	IRQP2F[2]	Output
	UART 1	82	52h	High level	IRQP2F[1]	Output
	CAN 1	83	53h	High level	IRQP2F[0]	Output
PL	FPGA [15:8]	91:84	5Bh:54h	High level	IRQF2P[15:8]	Input
SCU	Parity	92	5Ch	Rising edge	~	~
Reserved	~	95:93	5Fh:5Dh	~	~	~

7.2.4 Wait for Interrupt Event Signal (WFI)

The CPU can go into a wait state where it waits for an interrupt (or event) signal to be generated. The wait for interrupt signal that is sent to the PL is described in [Chapter 3, Application Processing Unit](#).

7.3 Register Overview

The ICC and ICD registers are part of the pl390 GIC register set. There are 60 SPI interrupts. This is far fewer than what the pl390 can support, so there are far fewer interrupt enable, status, prioritization and processor target registers in the ICD than is possible for the pl390. A summary of the ICC and ICD registers are listed in [Table 7-4](#)

Table 7-4: Interrupt Controller Register Overview

Name	Register Description	Write Protection Lockdown
Interrupt Controller CPU (ICC)		
ICCICR	CPU interface control	Yes, except EnableNS
ICCPMR	Interrupt priority mask	~
ICCBPR	Binary point for interrupt priority	~
ICCIAR	Interrupt acknowledge	~
ICCEOIR	End of interrupt	~
ICCRPR	Running priority	~
ICCHPIR	Highest pending interrupt	~
ICCABPR	Aliased non-secure binary point	~
Interrupt Controller Distributor (ICD)		
ICDDCR	Secure/non-secure mode select	Yes
ICDICTR, ICDIIDR	Controller implementation	~
ICDISR [2:0]	Interrupt security	Yes
ICDISER [2:0], ICDICER [2:0]	Interrupt set-enable and clear-enable	Yes
ICDISPR [2:0], ICDICPR [2:0]	Interrupt set-pending and clear-pending	Yes
ICDABR [2:0]	Interrupt active	~
ICDIPR [23:0]	Interrupt priority, 8-bit fields	Yes
ICDIPTR [23:0]	Interrupt processor targets, 8-bit fields	Yes
ICDICFR [5:0]	Interrupt sensitivity type, 2-bit field (level/edge, pos/neg)	Yes
PPI and SPI Status		
PPI_STATUS	PPI status: Corresponds to ICD*0 registers (security, enable, pending and active).	~
SPI_STATUS [2:1]	SPI status: Corresponds to ICD*{2:1} registers (security, enable, pending and active).	~
Software Generated Interrupts (SGI)		
ICDSGIR	Software-generated interrupts	~
Disable Write Accesses (SLCR register)		
APU_CTRL	CFGSDISABLE bit disables some write accesses	~

7.3.1 Write Protection Lock Down

The interrupt controller provides the facility to prevent write accesses to critical configuration registers. This is done by writing a one to the APU_CTRL[CFGSDISABLE] bit. The APU_CTRL register is part of the EPP's System Level Control register set, SLCR. This controls the write behavior for the secure interrupt control registers.

If the user wants to set the CFGSDISABLE bit, it is recommended that this be done during the user software boot process which occurs after the software has configured the Interrupt Controller registers. The CFGSDISABLE bit can only be cleared by a power-on reset (POR.) After the CFGSDISABLE bit is set, it changes the protected register bits to read-only and therefore the behavior of these secure interrupts cannot be changed, even in the presence of rogue code executing in the secure domain.

7.4 Programming Model

7.4.1 Interrupt Prioritization

All of the interrupt requests (PPI, SGI and SPI) are assigned a unique ID number. The controller uses the ID number to arbitrate. The interrupt distributor holds the list of pending interrupts for each CPU, and then selects the highest priority interrupt before issuing it to the CPU interface. Interrupts of equal priority are resolved by selecting the lowest ID.

The prioritization logic is physically duplicated to enable the simultaneous selection of the highest priority interrupt for each CPU. The interrupt distributor holds the central list of interrupts, processors and activation information, and is responsible for triggering software interrupts to the CPUs.

SGI and PPI distributor registers are banked to provide a separate copy for each connected processor. Hardware ensures that an interrupt targeting several CPUs can only be taken by one CPU at a time.

The interrupt distributor transmits to the CPU interfaces the highest pending interrupt. It receives back the information that the interrupt has been acknowledged, and can then change the status of the corresponding interrupt. Only the CPU that acknowledges the interrupt can end that interrupt.

7.4.2 Interrupt Handling

The response of the GIC to a pending interrupt when an IRQ line de-asserts is described in the ARM document: *IHI0048B_gic_architecture_specification.pdf* (see [Appendix A, Additional Resources](#)). See the Note in Section 1.4.2 with additional information in Section 3.2.4.

If the interrupt is pending in the GIC and IRQ is de-asserted, the interrupt in the GIC becomes inactive (and the CPU never sees it).

If the interrupt is active in the GIC (because the CPU interface has acknowledged the interrupt), then the software ISR determines the cause by checking the GIC registers first and then polling the I/O Peripheral interrupt status registers.

7.4.3 ARM Programming Topics

The ARM GIC architecture specification includes these programming topics:

- GIC register access
- Distributor and CPU Interfaces
- Affects of the GIC security extensions
- PU Interface registers
- Preserving and restoring controller state

7.4.4 Legacy Interrupts and Security Extensions

When the legacy interrupts (IRQ, FIQ) are used, and an interrupt handler accesses both IRQs and FIQs in secure mode (via ICCICR[AckCtl]=1), race conditions occasionally occur when reading the interrupt IDs. There is also a risk of seeing FIQ IDs in the IRQ handler, as the GIC only knows what security state the handler is reading from, not which type of handler.

There are two workable solutions:

- Only signal IRQs to a re-entrant IRQ handler and use the preemption feature in the GIC.
- Use FIQ and IRQ with ICCICR[AckCtl]=0 and use the TLB tables to handle IRQ in non-secure mode, and handle FIQ in secure mode.

Timers

8.1 Introduction

Each Cortex-A9 processor has its own private 32-bit timer and 32-bit watchdog timer. Both processors share a global 64-bit timer. These timers are always clocked at 1/2 of the CPU frequency (CPU_3x2x).

On the system-level, there is a 24-bit watchdog timer and two 16-bit triple timer/counters. The system watchdog timer is clocked at 1/4 or 1/6 of the CPU frequency (CPU_1x), or can be clocked by an external signal from an MIO pin or from the PL. The two triple timers/counters are always clocked at 1/4 or 1/6 of the CPU frequency (CPU_1x), and are used to count the widths of signal pulses from an MIO pin or from the PL.

8.1.1 System Diagram

The relationships of the system timers are shown in [Figure 8-1](#).

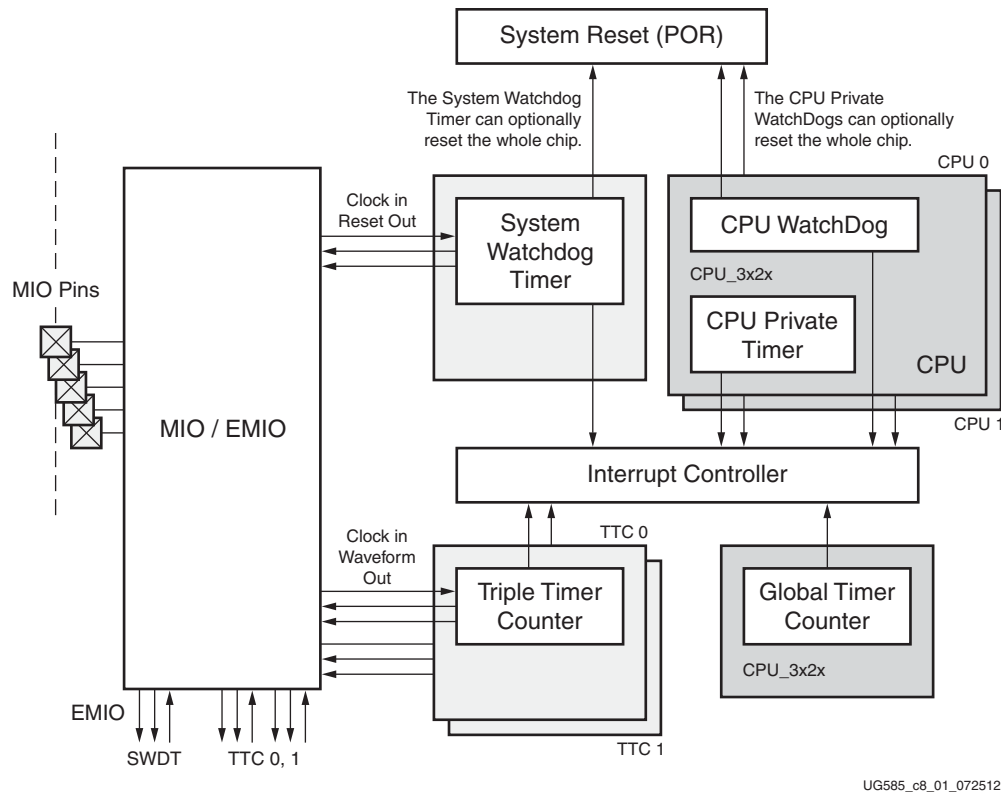


Figure 8-1: System View

8.2 CPU Private Timers and Watchdog Timers

The CPU private timers and watchdog timers are fully documented in the *Cortex-A9 MPCore Technical Requirements Document*, sections 4.1 and 4.2 (see [Appendix A, Additional Resources](#)).

Both the timer and watchdog blocks have the following features:

- 32-bit counter that generates an interrupt when it reaches zero
- 8-bit prescaler to enable better control of the interrupt period
- Configurable single-shot or auto-reload modes
- Configurable starting values for the counter
- SCU time interval is calculated as following:

$$\text{Timer Interval} = [(\text{Prescale_Val} + 1)(\text{Load_Val} + 1)] / \text{CPU_Peri_Clk}$$

Although peripheral logic only contributes a small portion of total Cortex-A9 MP logic, peripheral logic contributes a major portion of Cortex-A9 logic still up and running in sleep mode.

8.2.1 Clocking

All private timers and watchdog timers are always clocked at 1/2 of the CPU frequency (CPU_3x2x).

8.2.2 Register Overview

A register overview of the CPU private and watchdog timers is provided in in [Table 8-1](#).

Table 8-1: CPU Private Timers Register Overview

Function	Name	Overview
CPU Private Timers		
Reload and current values	Timer Load Timer Counter	Values to be reloaded into the decremter Current value of the decremter.
Control and Interrupt	Timer Control Timer Interrupt	Enable, Auto reload, IRQ, prescaler, interrupt status.
CPU Private Watchdogs (AWDT 0 and 1)		
Reload and current values	Watchdog Load Watchdog Counter	Values to be reloaded into the decremter. Current value of the decremter.
Control and Interrupt	Watchdog Control Watchdog Interrupt	Enable, Auto reload, IRQ, prescaler, interrupt status. (Note: cannot disable watchdog.)
Reset status	Watchdog Reset Status	Reset status as a result of watchdog reaching 0. Cleared with POR only, so SW can tell if the reset was caused by watchdog.
Disable	Watchdog Disable	Disable watchdog via a sequence of writes of two specific words.

8.3 Global Timer (GT)

The Global Timer is fully documented in the *Cortex-A9 MPCore Technical Requirements Document*, sections 4.3 and 4.4 (see [Appendix A, Additional Resources](#)). The global timer is a 64-bit incrementing counter with an auto-incrementing feature. The global timer is memory-mapped in the same address space as the private timers. The global timer is accessed at reset in secure state only. The global timer is accessible to all Cortex-A9 processors. Each Cortex-A9 processor has a 64-bit comparator that is used to assert a private interrupt when the global timer has reached the comparator value.

8.3.1 Clocking

The GTC is always clocked at 1/2 of the CPU frequency (CPU_3x2x).

8.3.2 Register Overview

A register overview of the GTC is provided in [Table 8-2](#).

Table 8-2: Global Timer Register Overview

Function	Name	Overview
Global Timer (GTC)		
Current values	Global Timer Counter	Current value of the incrementer
Control and Interrupt	Global Timer Control Global Interrupt	Enable timer, enable comparator, IRQ, auto-increment, interrupt status
Comparator	Comparator Value Comparator Increment	Current value of the comparator Increment value for the comparator
	Global Timer Disable	Disable watchdog via a sequence of writes of two specific words

8.4 System Watchdog Timer (SWDT)

In addition to the two CPU private watchdog timers, there is a system watchdog timer (SWDT) for signaling additional catastrophic system failure, such as a PS PLL failure. Unlike the AWDT, the SWDT can run off the clock from an external device or the PL, and provides a reset output to an external device or the PL.

8.4.1 Features

Key features of the available timers/counters are as follows:

- An internal 24-bit counter
- Selectable clock input from:
 - Internal PS bus clock (CPU_1x)
 - Internal clock (from PL)
 - External clock (from MIO)
- On time out, outputs one or a combination of:
 - System interrupt (PS)
 - System reset (PS, PL, MIO)
- Programmable time out period:
 - Time out range 32,760 to 268,431,360 clock cycles (330 μ s to 2.7s at 100 MHz)
- Programmable output signal duration on time out:
 - System interrupt pulse 4 to 32 clock cycles (40 ns to 320 ns at 100 MHz)
 - System reset pulse 2 to 256 clock cycles (20 ns to 2.6 μ s at 100 MHz)

8.4.2 Block Diagram

A block diagram of the SWDT is shown in Figure 8-2.

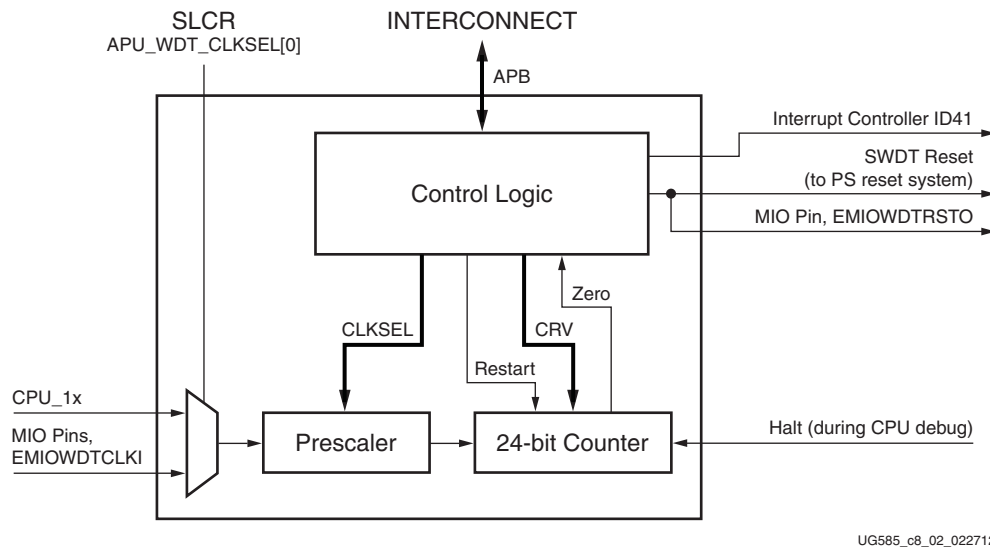


Figure 8-2: System Watchdog Timer Block Diagram

Notes relevant to Figure 8-2:

- SLCR programmable registers (APU_WDT_CLKSEL, MIO control) select the clock input.
- SWDT programmable registers set the values for CLKSEL and CRV.
- Signal *restart* causes the 24-bit counter to reload the CRV values, and restart counting.
- Signal *halt* causes the counter to halt during CPU debug (same behavior as AWDT).

8.4.3 Functional Description

The control logic block has an APB interface connected to the system interconnect. Each write data received from APB has a key field which must match the key of the register to be written in order to be able to write to the register.

The Zero Mode register controls the behavior of the SWDT when its internal 24-bit counter reaches zero. Upon receiving a zero signal, the control logic block asserts the interrupt output signal for IRQLN clock cycles if both WDEN and IRQEN are set, and also asserts the reset output signals for RSTLN clock cycles if both WDEN and RSTLN are set. The 24-bit counter then stays at zero until it is restarted.

The Counter Control register sets the time out period, via setting reload values in `swdt.CONTROL[CLKSET]` and `swdt.CONTROL[CRV]` to control the prescaler and the 24-bit counter.

The Restart register is used to restart the counting process. Writing to this register with a matched key causes the prescaler and the 24-bit counter to reload the values from CRV signals.

The Status register shows whether the 24-bit counter reaches zero. Regardless of the WDEN bit in the Zero Mode register, the 24-bit counter always keeps counting down to zero if it is not zero and the selected clock source is present. Once it reaches zero, the WDZ bit of the Status register is set and remains set until the 24-bit counter is restarted.

The prescaler block divides down the selected clock input. The CLKSEL signal is sampled at every rising clock edge.

The internal 24-bit counter counts down to zero and stays at zero until it is restarted. While the counter is at zero, the zero output signal is High.

8.4.4 Register Overview

A register overview of the SWDT is provided in in [Table 8-3](#).

Table 8-3: System Watchdog Timer Register Overview

Function	Name	Overview
Zero mode	swdt.MODE	Enable SWDT, enable interrupt and reset outputs on time out, set output pulse lengths.
Reload values	swdt.CONTROL	Set the reload values for prescaler and 24-bit counter on time out.
Restart	swdt.RESTART	Cause the prescaler and the 24-bit counter to reload and restart.
Status	swdt.STATUS	Indicates watchdog reaching zero.

8.4.5 Programming Model

System Watchdog Timer Enable Sequence

1. Select clock input source (SLCR[WDT_CLK_SEL[SEL]] bit):
Ensure that the SWDT is disabled (SWDT[MODE[WDEN]] is 0) before proceeding with this step. Changing the clock input source when the SWDT is enabled results in unpredictable behavior.
2. Set the time out period (Counter Control register):
The SWDT[CONTROL[CKEY]] field must be 0x248 in order to be able to write this register.
3. Enable the counter; enable output pulses; set up output pulse lengths (Zero Mode register):
The SWDT[MODE[ZKEY]] field must be 0xAB0 in order to be able to write this register. Ensure that IRQLN and RSTLN meet the specified minimum values.
4. To run the SWDT with a different setting, disable the timer first (SWDT[MODE[WDEN]] bit). Then repeat the above steps.

8.5 Triple Timer Counters (TTC)

The TTC contains three independent timers/counters. Because a single APB interface is used to access them, the three timers/counters must have the same security status. There are two TTC modules in the PS, for a total of six timers/counters. One TTC is reserved for secure software and the other is shared by either secure software or non-secure software. When TrustZone is not used, both TTC modules are available to non-secure software.

8.5.1 Features

Each of the triple timer counters has:

- Three independent 16-bit prescalers and 16-bit up/down counters
- Selectable clock input from:
 - Internal PS bus clock (CPU_1x)
 - Internal clock (from PL)
 - External clock (from MIO)
- Three interrupts, one for each counter
- Interrupt on overflow, at regular interval, or counter matching programmable values
- Generates waveform output (e.g., PWM) through the MIO and to the PL

8.5.2 Block Diagram

A block diagram of the TTC is shown in [Figure 8-3](#). The clock-in and wave-out multiplexing for Timer/Clock 0 is controlled by the slcr.PIN_MUX registers. If no selection is made in these registers, then the default becomes the EMIO interface.

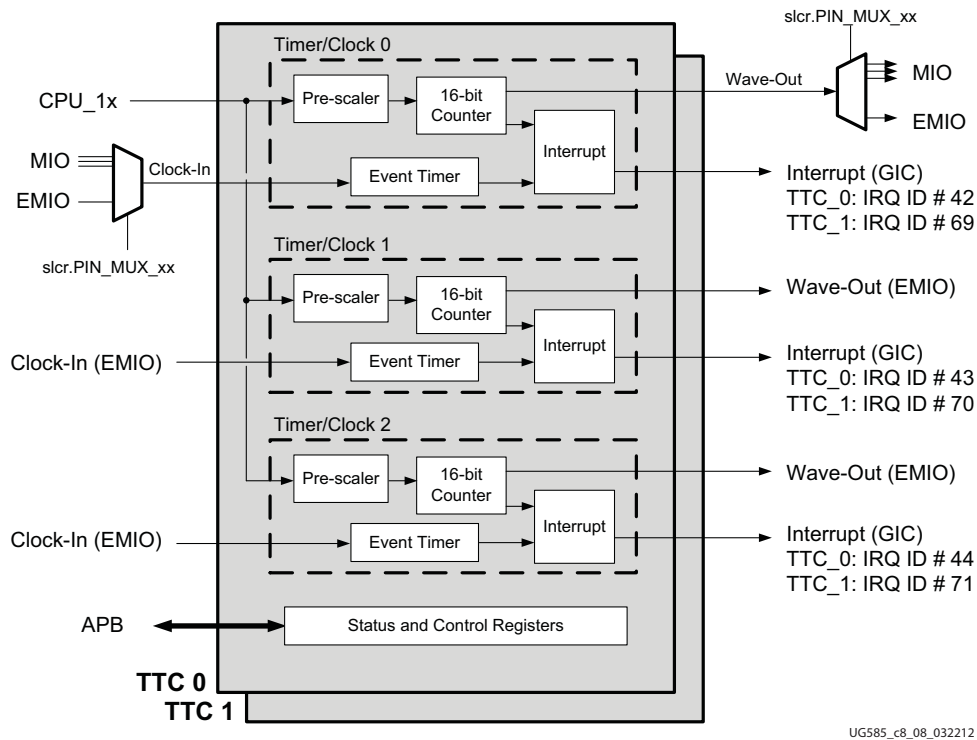


Figure 8-3: Triple Counter Timer Block Diagram

8.5.3 Functional Description

Each prescaler module can be independently programmed to use the PS internal bus clock (CPU_1x), or an external clock from the MIO or the PL. For an external clock, SLCR registers determine the exact pinout via the MIO or from the PL. The selected clock is then divided down from /2 to /65536, before being applied to the counter.

The counter module can count up or count down, and can be configured to count for a given interval. It also compares three match registers to the counter value, and generate an interrupt if one matches.

The interrupt module combines interrupts of various types: counter interval, counter matches, counter overflow, event timer overflow. Each type can be individually enabled.

Modes of Operation

Each counter module can be independently programmed to operate in either of the following two modes:

Interval mode: The counter increments or decrements continuously between 0 and the value of the Interval register, with the direction of counting determined by the DEC bit of the Counter Control register. An Interval interrupt is generated when the counter passes through zero. The corresponding Match interrupt is generated when the counter value equals one of the Match registers.

Overflow mode: The counter increments or decrements continuously between 0 and 0xFFFF, with the direction of counting determined by the DEC bit of the Counter Control register. An Overflow interrupt is generated when the counter passes through zero. The corresponding Match interrupt is generated when the counter value equals one of the Match registers.

Event Timer Operation

The event timer operates by having an internal (invisible to users) 16-bit counter clocked at CPU_1x which:

- Resets to 0 during the non-counting phase of the external pulse
- Increments during the counting phase of the external pulse

The Event Control Timer register controls the behavior of the internal counter:

- **E_En bit:** When 0, immediately resets the internal counter to 0, and stops incrementing
- **E_Lo bit:** Specifies the counting phase of the external pulse
- **E_Ov bit:** Specifies how to handle overflow at the internal counter (during the counting phase of the external pulse)
 - When 0: Overflow causes E_En to be 0 (see E_En bit description)
 - When 1: Overflow causes the internal counter to wrap around and continues incrementing
 - An interrupt is always generated (subject to further enabling through another register) when an overflow occurs

The Event register is updated with the non-zero value of the internal counter at the end of the counting-phase of the external pulse; therefore, it shows the widths of the external pulse, measured in number of cycles of CPU_1x.

If the internal counter is reset to 0, due to overflow, during the counting phase of the external pulse, the Event register will not be updated and maintains the old value from the last non-overflowing counting operation.

8.5.4 Register Overview

A register overview of the TTC is provided in in [Table 8-4](#).

Table 8-4: Triple Timer Counter Register Overview

Function	Name	Overview
Clock Control	Clock Control Register	Controls prescaler, selects clock input, edge
	Counter Control Register	Enables counter, sets mode of operation, sets up/down counting, enables matching, enables waveform output
Status	Counter Value Register	Returns current counter value

Table 8-4: Triple Timer Counter Register Overview (Cont'd)

Function	Name	Overview
Counter Control	Interval Register	Sets interval value
	Match Register 1	Sets match values, total 3
	Match Register 2	
	Match Register 3	
Interrupt	Interrupt Register	Shows current interrupt status
	Interrupt Enable Register	Enable interrupts
Event	Event Control Timer Register	Enable event timer, stop timer, sets phrase
	Event Register	Shows width of external pulse

8.5.5 Programming Model

Counter Enable Sequence

1. Select clock input source, set prescaler value (slcr.MIO_MUX_SEL registers, TTC Clock Control register): Ensure TTC is disabled (TTC[Counter_Control_n [DIS]] is 1) before proceeding with this step.
2. Set interval value (Interval register): This step is optional, for interval mode only.
3. Set match value (Match registers): This step is optional, if matching is to be enabled.
4. Enable interrupt (Interrupt Enable register): This step is optional, if interrupt is to be enabled.
5. Enable/disable waveform output, enable/disable matching, set counting direction, set mode, enable counter (TTC Counter Control register): This step starts the counter.

Counter Stop Sequence

1. Read back the value of Counter Control register.
2. Set DIS bit to 1, while keeping other bits.
3. Write back to Counter Control register.

Counter Restart Sequence

1. Read back the value of Counter Control register.
2. Set RST bit to 1, while keeping other bits.
3. Write back to Counter Control register.

Event Timer Enable Sequence

1. Select external pulse source (slcr.MIO_MUX_SEL registers): The width of the selected external pulse is measured in CPU_1x period.

2. Set overflow handling, select external pulse level, enable the event timer (Event Control Timer register): This step starts measuring the width of the selected level (High or Low) of the external pulse.
3. Enable interrupt (Interrupt Enable register): This step is optional, if interrupt is to be enabled.
4. Read the measured width (Event register): Note that the returned value is not correct when overflow happened. See the description for the E_Ov bit of the Event Control Timer register in section 8.5.3 Functional Description.

Interrupt Clear and Acknowledge Sequence

1. Read Interrupt register: All bits in the Interrupt register are cleared on read.

8.5.6 I/O Signals

Timer I/O signals are identified in Table 8-5. The MIO pins and any restrictions based on device version are shown in the MIO table in section 2.4.4 MIO-at-a-Glance Table.

There are two triple timer counters (TTC0 and TTC1) in the system. Each TTC has three sets of interface signals: clock in and wave out for counter/timers 0, 1, and 2.

For each triple timer counter, the signals for counter/timer 0 can be routed to the MIO using the MIO_PIN registers. If the clock in or wave out signal is not selected by the MIO_PIN register, then the signal is routed to EMIO by default.

The signals for counter/timers 1 and 2 are only available through the EMIO.

Table 8-5: TTC I/O Signals

TTC	Timer Signal	I/O	MIO Pins	EMIO Signals	Controller Default Input Value
TTC0	Counter/Timer 0 Clock In	I	19, 31, 43	EMIOTTC0CLKI0	0
	Counter/Timer 0 Wave Out	O	18, 30, 42	EMIOTTC0WAVEO0	~
	Counter/Timer 1 Clock In	I	N/A	EMIOTTC0CLKI1	0
	Counter/Timer 1 Wave Out	O	N/A	EMIOTTC0WAVEO1	~
	Counter/Timer 2 Clock In	I	N/A	EMIOTTC0CLKI2	0
	Counter/Timer 2 Wave Out	O	N/A	EMIOTTC0WAVEO2	~
TTC1	Counter/Timer 0 Clock In	I	17, 29, 41	EMIOTTC1CLKI0	0
	Counter/Timer 0 Wave Out	O	16, 28, 40	EMIOTTC1WAVEO0	~
	Counter/Timer 1 Clock In	I	N/A	EMIOTTC1CLKI1	0
	Counter/Timer 1 Wave Out	O	N/A	EMIOTTC1WAVEO1	~
	Counter/Timer 2 Clock In	I	N/A	EMIOTTC1CLKI2	0
	Counter/Timer 2 Wave Out	O	N/A	EMIOTTC1WAVEO2	~

System watchdog timer I/O signals are identified in Table 8-6.

Table 8-6: Watchdog Timer I/O Signals

SWDT Signal	I/O	MIO Pins	EMIO Signals	Controller Default Input Value
Clock In	I	14, 26, 38, 50, 52	EMIOWDTCCLKI	0
Reset Out	O	15, 27, 39, 51, 53	EMIOWDTRSTO	~

DMA Controller

9.1 Introduction

The DMA controller (DMAC) uses an AXI master interface to perform DMA data transfers to/from system memories and peripherals. The DMAC includes a small instruction set that provides a flexible method of specifying the DMAC operations. This provides greater flexibility than the fixed capabilities of linked-list item (LLI)-based DMA controllers. The program code is stored in a region of system memory that the DMAC accesses using its AXI master interface. The DMAC can be configured with up to eight DMA channels with each channel capable of supporting a single concurrent thread of DMA operation. The DMAC has a single DMA manager that can be used to initialize DMA channel threads. When a DMA channel thread executes a load or store instruction, the DMAC adds the instruction to the relevant read or write queue.

The DMAC uses these queues as an instruction storage buffer prior to issuing the instructions on the AXI bus. The DMAC also contains a multichannel first-in-first-out (MFIFO) data buffer to store data read or written to during a DMA transfer. To a program, it appears as a set of variable-depth parallel FIFOs, one per channel, with the restriction that the total depth of all of the FIFOs cannot exceed the size of the MFIFO.

The DMAC is able to move data from memory to memory without processor intervention. The data are transferred using only the AXI master interface. The source and destination memory could be anywhere in the system (PS or PL).

Another use case is to be able to move data from memory to peripheral on the PL and from peripheral in the PL to memory. The memory could be anywhere in the system. This is performed using the AXI interface and one of the four peripheral request interfaces. This peripheral request interface is necessary to control the data flow without processor intervention.

There are no peripheral request interfaces directed to the I/O Peripherals (IOP) in the PS. Processor intervention is needed to avoid underflow or overflow of the FIFOs in the targeted PS peripheral.

Dual APB interfaces enable the operation of the DMAC to be partitioned into the secure state and non-secure state. The APB interfaces can be used to access status registers and also directly execute instructions in the DMAC.

The DMAC does not support secure/non-secure channel mixing. The entire DMAC can be configured as either secure or non-secure. Security configuration changes are controlled by system-level control registers, and require a DMAC reset to take effect.

The DMAC is an Advanced Microcontroller Bus Architecture (AMBA) PrimeCell peripheral that is developed, tested, and licensed by ARM. Refer to the *ARM PrimeCell DMA Controller (PL330) Technical Reference Manual* for more details (see [Appendix A, Additional Resources](#)).

9.1.1 Features, Constraints and Limitations

The DMAC provides these features:

- Supports sixteen interrupt and event signals
 - Eight interrupts to the processors and the PL (parallel outputs going to both destinations)
 - Eight internal events
- An instruction set that provides flexibility for programming DMA transfers
- 128 64-bit words MFIFO to buffer the data that the controller writes or reads during a transfer
- Supports eight cache lines and each cache line is four deep
- Supports eight concurrent DMA channels
 - Allows multiple threads to execute in parallel
 - Support eight read issuing capability
 - Support eight write issuing capability
- Supports security
 - Dedicated APB slave interface for secure register accessing
 - Entire DMAC can be configured as either secure or non-secure
- Supports memory-to-memory transfer type
 - Up to 1 GB/s DDR memory-to-memory copy in heavy load environment
- Supports memory-to-PL peripheral and PL peripheral-to-memory transfer type
 - Four peripheral request interfaces (to the PL) without CPU intervention
 - Each interface accepts up to four active requests
- Supports Scatter-gather type of transfers

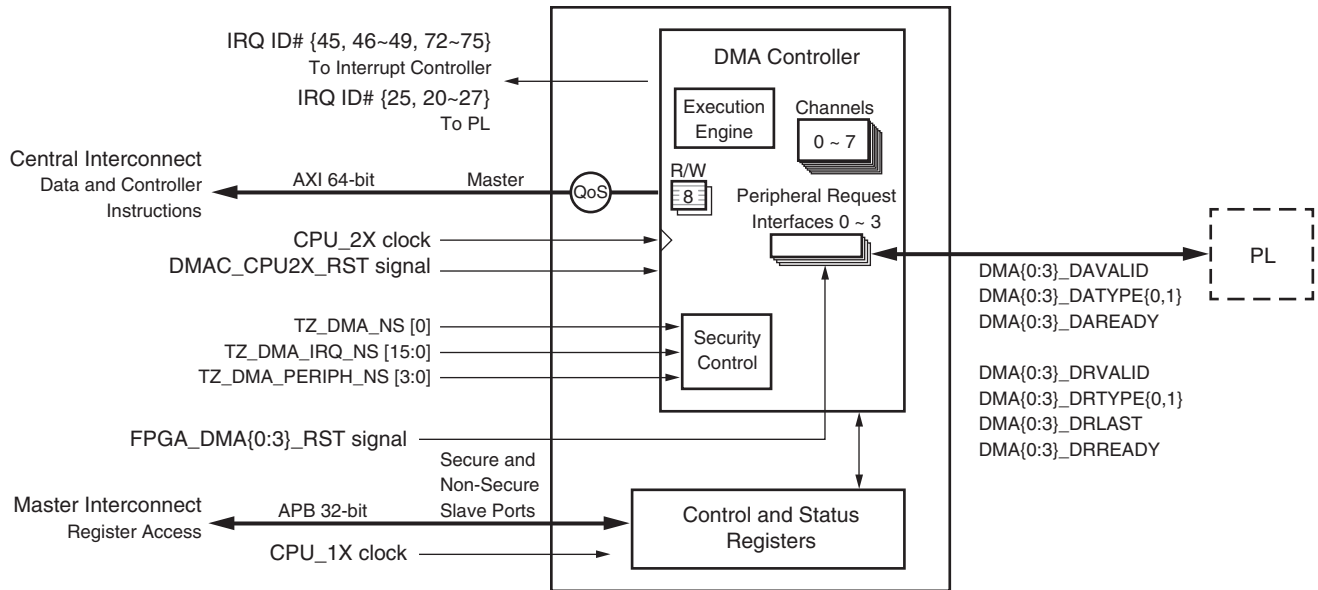
The DMAC has the following constraints and limitations:

- Channel arbitration
 - Round-robin scheme to service the active DMA channels
 - Services the DMA manager prior of servicing the next DMA channel
 - Changes to the arbitration process are not supported
- Channel prioritization
 - Responds to all active DMA channels with equal priority
 - Changes to the priority of a DMA channel over any other DMA channels are not supported
- Instruction cache latency
 - When a cache miss occurs, the latency to service the request is mainly dependent on the read latency of the AXI bus. The latency that the DMAC adds is minimal.
- AXI data transfer size
 - Performs data accesses up to the 64-bit width of the AXI data bus

- Signals a precise abort if the user programs the `src_burst_size` or `dst_burst_size` fields to be larger than 64 bits
- AXI bursts crossing 4 KB boundaries
 - The AXI specification does not permit AXI bursts to cross 4 KB address boundaries
 - If the DMAC is programmed with a combination of burst start address, size, and length that would cause a single burst to cross a 4 KB address boundary, then the DMAC instead generates a pair of bursts with a combined length equal to that specified. This operation is transparent to the DMAC channel thread program so that, for example, the DMAC responds to a single DMALD instruction by generating the appropriate pair of AXI read bursts.
- AXI burst types
 - Can be programmed to generate only fixed-address or incrementing-address burst types for data accesses. Wrapping-address bursts are not generated for data accesses or for instruction fetches.
- AXI write addresses
 - Can issue multiple outstanding write addresses up to eight (write issuing capability)
 - The DMAC does not issue a write address until it has read in all of the data bytes required to fulfill that write transaction.
- AXI write data interleaving
 - Does not generate interleaved write data. All write data beats for one write transaction are output before any write data beat for the next write transaction.
- AXI characteristics
 - Does not support locked or exclusive accesses

9.1.2 System Viewpoint

The system viewpoint of the DMA controller is shown in Figure 9-1.



UG585_c9_01_031212

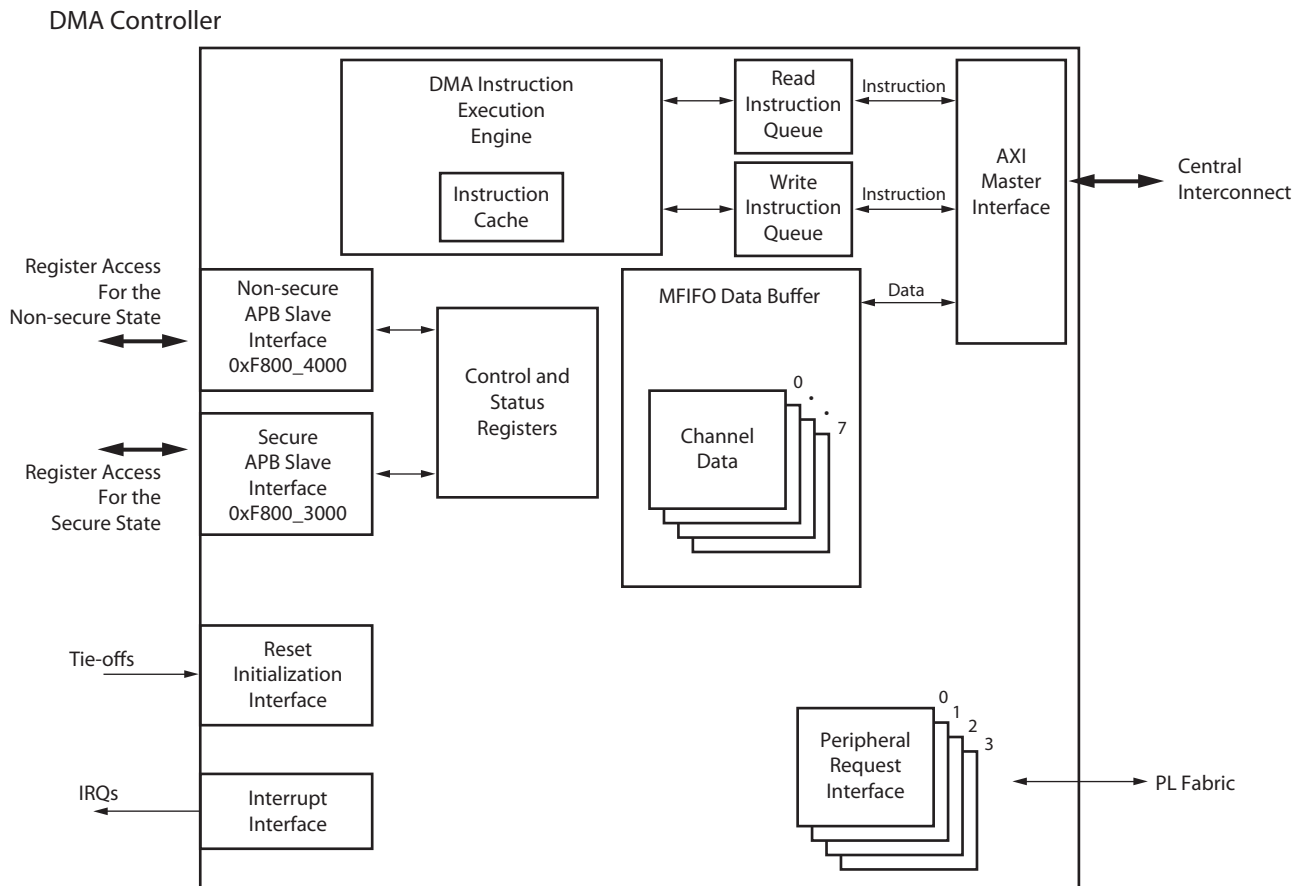
Figure 9-1: DMA Controller System Viewpoint

Clocks and Resets

See Chapter 25, Clocks and Chapter 26, Reset System.

9.1.3 Block Diagram

The block diagram of the DMA controller is shown in Figure 9-2.



UG_585_c9_02_030712

Figure 9-2: DMA Controller Block Diagram

DMA Instruction Execution Engine (Operating States)

The DMAC contains an instruction processing block that enables it to process program code that controls a DMA transfer. The DMAC maintains a separate state machine for each thread.

Instruction Cache

The DMAC stores instructions temporarily in a cache. When a thread requests an instruction from an address, the cache performs a look-up. If a cache hit occurs then the cache immediately provides the data, otherwise the thread is stalled while the DMAC uses the AXI interface to perform a cache line fill. If an instruction is greater than four bytes, or spans the end of a cache line, then it performs multiple cache accesses to fetch the instruction.

Note: When a cache line fill is in progress, the DMAC enables other threads to access the cache, but if another cache miss occurs the pipeline is stalled until the first line fill is complete.

Read/Write Instruction Queues

When a DMA channel thread executes a load or store instruction the DMAC adds the instruction to the relevant read queue or write queue. The DMAC uses these queues as an instruction storage buffer prior to issuing the instructions on the AXI central interconnect.

Multi-channel Data FIFO

The DMAC uses a multi-channel first-in-first-out (MFIFO) data buffer to store data that it reads, or writes, during a DMA transfer.

AXI Master Interface

The program code is stored in a region of system memory that the DMAC accesses using this AXI master interface. The AXI master interface also enables the DMA to transfer data from a source AXI slave to a destination AXI slave.

APB Slave Interfaces

The DMAC provides these APB interfaces:

- Non-secure APB slave interface
- Secure APB slave interface

The APB slave interface connects the DMAC to the APB and enables a microprocessor to access the registers.

Control and Status Registers

Using these registers, a microprocessor can:

- Access the status of the DMA manager
- Access the status of the DMA channel threads
- Enable or clear interrupts
- Enable events
- Issue an instruction for the DMAC to execute

Peripheral Request Interfaces

The peripheral request interface supports the connection of DMA-capable peripherals resident in the PL. Each peripheral request interface is asynchronous to one another and asynchronous to the DMA itself.

Interrupt Interface

The interrupt interface enables efficient communications of events to the interrupt controller.

Reset Initialization Interface

This interface enables the user to initialize the operating state of the DMAC as it exits from reset. When the DMAC exits from reset:

- It reads the status of the SLCR register TZ_DMA_NS to set the security of the DMA manager.
- It reads the status of the SLCR register TZ_DMA_PERIPH_NS to enable assignment of each peripheral request interface to a security state.
- It reads the status of the SLCR register TZ_DMA_IRQ_NS signals to enable assignment of each irq[x] signal to a security state.
- It waits for an instruction from either APB interface.

Note: When set, each security state remains constant until the DMAC resets.

Refer to [External Signals](#) for detailed description of this interface.

9.2 Functional Description

All of the DMA transactions use the AXI master interface to move data. If a DMA-capable peripheral in the PL is involved in the transaction, one of the four peripheral request interfaces must control the flow of data on the AXI to avoid overflow or underflow issues. Therefore, there are two categories of DMA transactions:

- Memory to memory transaction. The only interface involved is the AXI interface. An example of dataflow from OCM to DDR is shown in [Figure 9-3](#).
- Memory to/from PL transaction. On top of the AXI interface, one of the four peripheral request interfaces controls the data flow. [Figure 9-4](#) shows the data flow from a peripheral in the PL to the DDR. The peripheral request interface is not shown in the figure because it is only used to control the flow and relies on the AXI interface to move data.

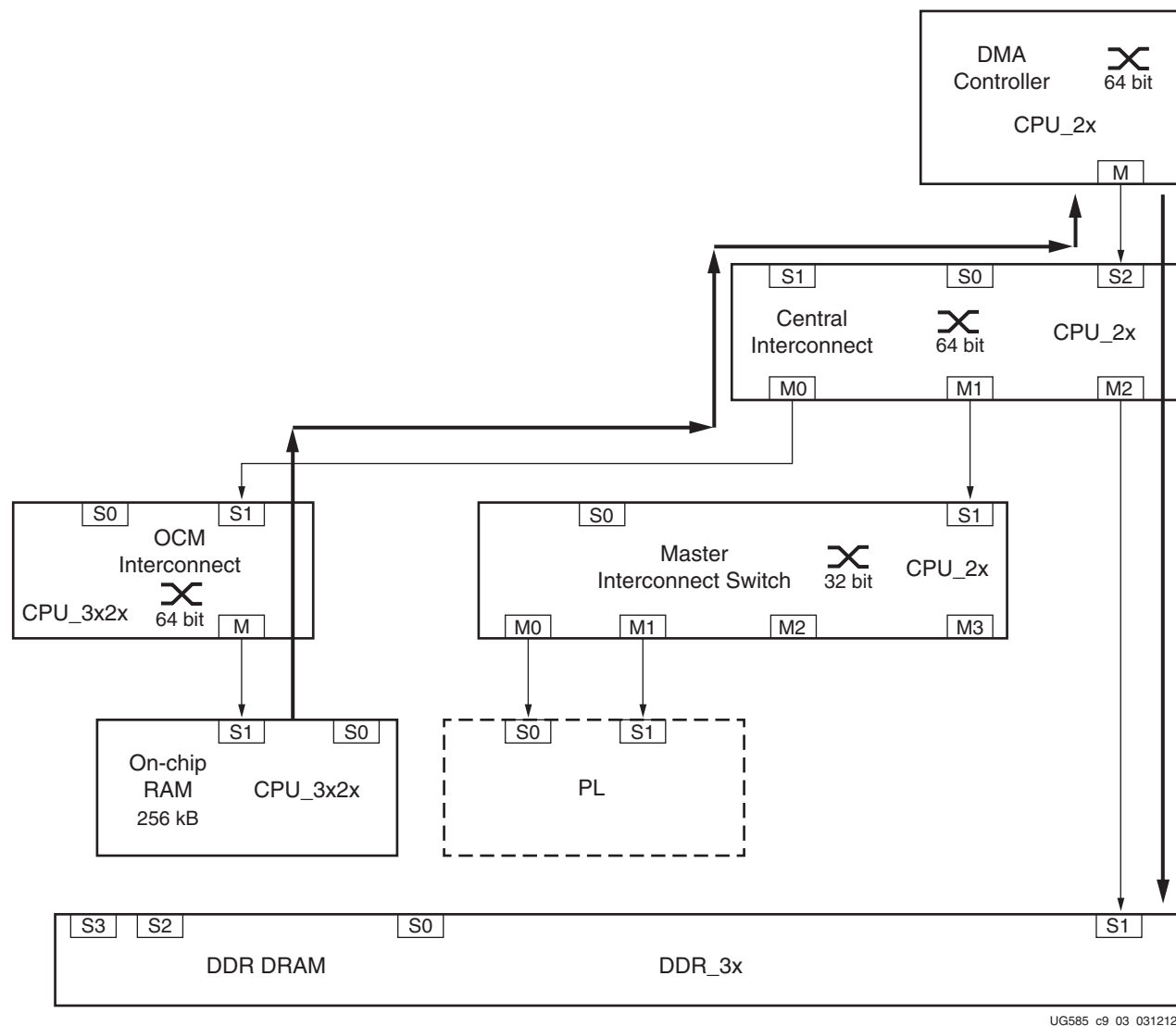


Figure 9-3: Example of Data Flow During an OCM to DDR Transfer

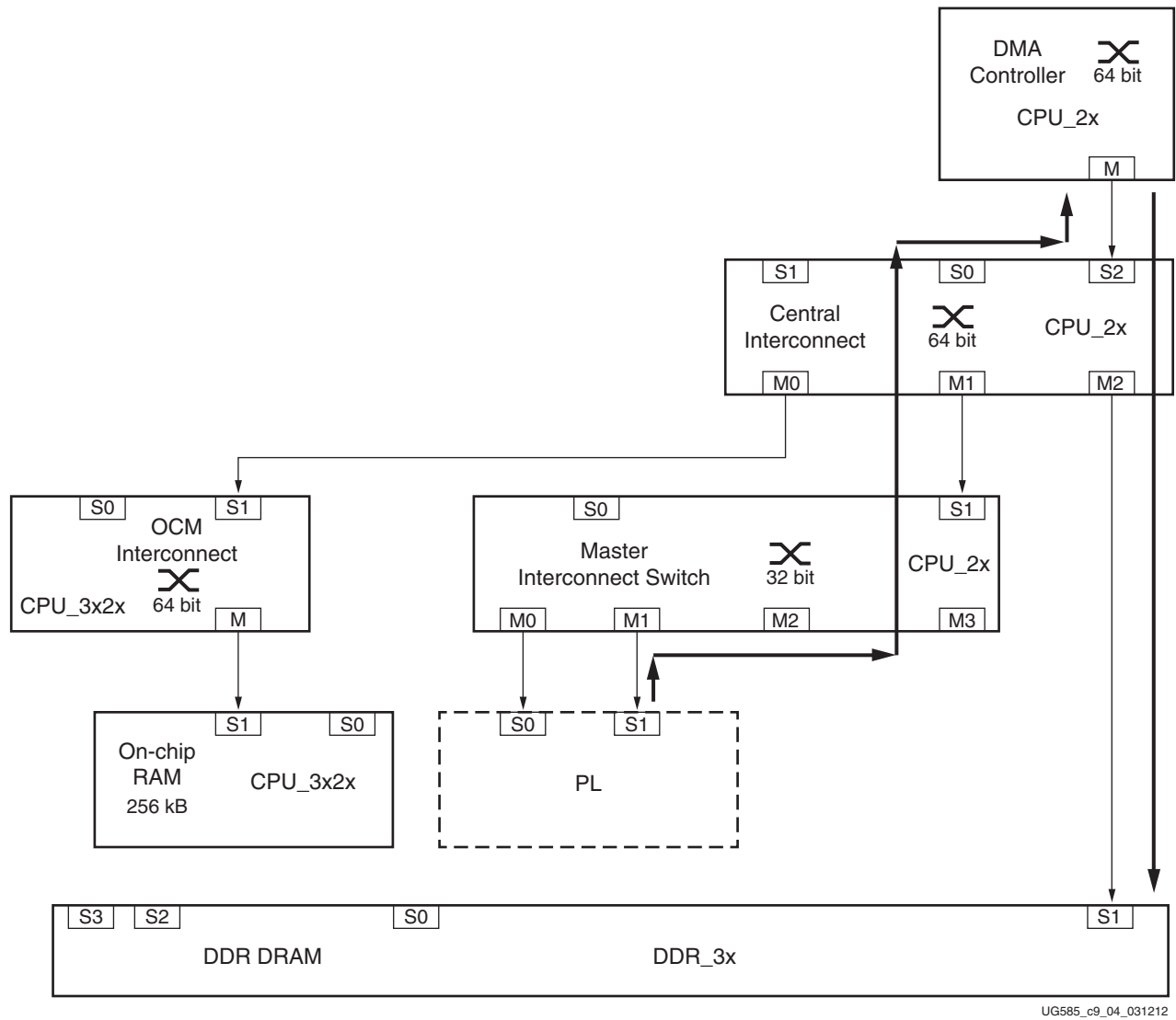


Figure 9-4: Example of Dataflow During a PL Peripheral to DDR Transfer

9.2.1 Programming the DMA Controller

This section describes how to issue instructions to the DMA manager using one of the two APB interfaces available.

When the DMAC is operating in real time, the user can only issue the following limited subset of instructions:

DMAGO	Starts a DMA transaction using a DMA channel that the user specifies.
DMASEV	Signals the occurrence of an event, or interrupt, using an event number that the user specifies.
DMAKILL	Terminates a thread.

The appropriate APB interface must be used depending on the security state in which the SLCR register TZ_DMA_NS initializes the DMA manager to operate. For example, if the DMA manager is in the secure state, the instruction using the secure APB interface must be used or the DMAC ignores the instruction. The non-secure APB interface is the suggested port to use to start or restart a DMA channel when the DMA manager is in the non-secure state, however, the secure APB interface can be used in non-secure mode. (See [Security Usage](#) for more details.)

Before issuing instructions using the Debug Instruction registers or the DBGCMD register, the DBGSTATUS register must be read to ensure that debug is idle, otherwise, the DMA manager ignores the instructions. See the Debug Command register and Debug Status register in [Appendix B, Register Details](#).

When the DMA manager receives an instruction from an APB slave interface, it can take several clock cycles before it can process the instruction — for example, if the pipeline is busy processing another instruction.

Prior to issuing DMAGO, the system memory must contain a suitable program for the DMA channel thread to execute, starting at the address that the DMAGO specifies.

Refer to [Programming Examples Reference](#) for a detailed step-by-step procedure to start a DMA channel thread using the debug instruction registers.

9.2.2 Memory to Memory Transaction

The controller uses a master on the AXI central interconnect switch to access memories in the system, such as:

- OCM
- DDR

Through the same AXI central interconnect the controller can also access the majority of the peripheral subsystems. If the data of a target peripheral can be seen as a memory-mapped region, the memory to memory transaction can be used. In this kind of transaction the DMAC does not have to handle the peripheral FIFOs and there are no overflow and underflow issues. Typical examples are:

- QSPI in Linear addressing mode
- NOR flash
- NAND flash

The memory map for the DMA controller is shown in [Chapter 4, System Addresses](#).

For more information on the AXI Interfaces, refer to [AXI Master Interface](#). Examples of memory to memory transfer are provided in [Programming Examples Reference](#).

9.2.3 Memory to/from PL Peripheral Transaction

The majority of peripherals allow transferring data through FIFOs. These FIFOs must be managed to avoid overflow and underflow situations. For this reason four specific peripheral request interfaces are available to connect the DMAC to DMA-capable peripherals in the PL. Each one of these interfaces can be assigned to any DMA channel.

The DMAC is configured to accept up to four active requests for each peripheral interface. An active request is where the DMAC has not started the requested AXI data transfers. The DMAC has a request FIFO for each peripheral interface, which it uses to capture the requests from a peripheral. When a request FIFO is full, the DMAC sets the corresponding DMA{3:0}_DRREADY Low to signal that the DMAC cannot accept any requests sent from the peripheral.

There are no peripheral request interfaces directed to the I/O peripherals (IOP) in the PS. Processor intervention is needed to avoid underflow or overflow of the FIFOs in the targeted PS peripheral.

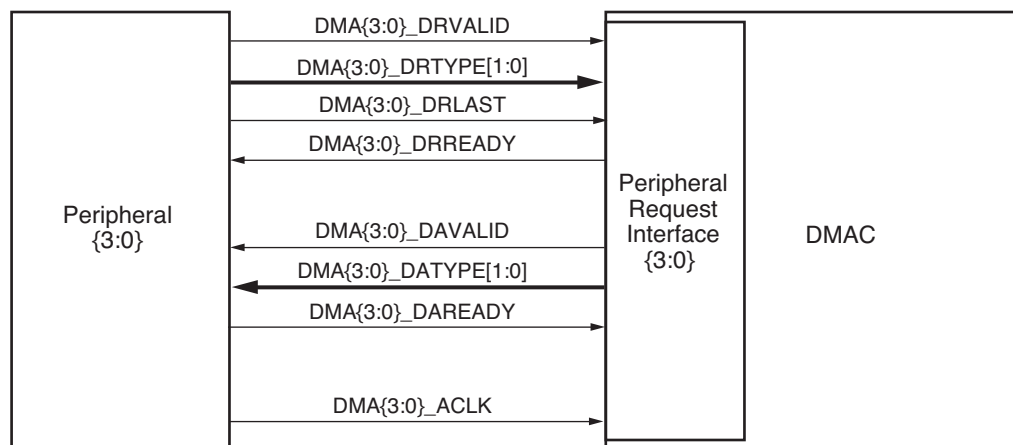
There are two different way to handle the quantity of the data flowing between the DMAC and the peripheral:

Peripheral length management:	The peripheral controls the quantity of data that is contained in a DMA cycle.
DMAC length management:	The DMAC is controlling the quantity of data in a DMA cycle.

Peripheral Request Interface

Figure 9-5 shows that the peripheral request interface consists of a peripheral request bus and a DMAC acknowledge bus that use the prefixes:

DR	Peripheral request bus
DA	DMAC acknowledge bus



UG585_c9_05_030312

Figure 9-5: Request and Acknowledge Buses on the Peripheral Request Interface

Both buses use the valid-ready handshake that the AXI protocol describes. For more information on the handshake process, see the *AMBA AXI Protocol v1.0 Specification*.

The peripheral uses DMA{3:0}_DRTYPE[1:0] to:

- Request a single transfer
- Request a burst transfer
- Acknowledge a flush request

The DMAC uses DMA{3:0}_DATYPE[1:0] to:

- Signal when it completes the requested single transfer
- Signal when it completes the requested burst transfer
- Issue a flush request

The peripheral uses DMA{3:0}_DRLAST to:

- Signal the DMAC when the last DMA transfer commences

Handshake Rules

The DMAC uses the DMA handshake rules that [Table 9-1](#) shows, when a DMA channel thread is active, that is, not in the Stopped state. See [Peripheral Request Interface Timing Diagram](#) for more information.

Table 9-1: Handshake Rules

Rule	Description ⁽¹⁾
1	DMA{3:0}_DRVALID can change from Low to High on any DMA{3:0}_ACLK cycle, but must only change from High to Low when DMA{3:0}_DRREADY is High.
2	DMA{3:0}_DRTYPE can only change when either: <ul style="list-style-type: none"> • DMA{3:0}_DRREADY is High • DMA{3:0}_DRVALID is Low
3	DMA{3:0}_DRLAST can only change when either: <ul style="list-style-type: none"> • DMA{3:0}_DRREADY is High • DMA{3:0}_DRVALID is Low
4	DMA{3:0}_DAVALID can change from Low to High on any DMA{3:0}_ACLK cycle, but must only change from High to Low when DMA{3:0}_DAREADY is High
5	DMA{3:0}_DATYPE can only change when either: <ul style="list-style-type: none"> • DMA{3:0}_DAREADY is High • DMA{3:0}_DAVALID is Low

Notes:

1. All signals are only permitted to change state when DMA{3:0}_ACLK changes state.

Mapping to a DMA Channel

The DMAC enables the user to assign a peripheral request interface to any of the DMA channels. When a DMA channel thread executes DMAWFP, the value programmed in the peripheral [4:0] field

specifies the peripheral associated with that DMA channel. See DMAWFP instruction in [Instruction Set Reference for Manager and Commands](#).

PL Peripheral Length Management

The peripheral request interface enables a peripheral to control the quantity of data that a DMA cycle contains, without the DMAC being aware of how many data transfers it contains. The peripheral controls the DMA cycle by using:

DMA{3:0}_DRTYPE[1:0]	Selects a single or burst transfer
DMA{3:0}_DRLAST	Notifies the DMAC when it commences the final request in the current series

When the DMAC executes a DMAWFP instruction, it halts execution of the thread and waits for the peripheral to send a request. When the peripheral sends the request, the DMAC sets the state of the request flags depending on the state of the following signals:

DMA{3:0}_DRTYPE[1:0]	The DMAC sets the state of the request_type flag:	
DMA{3:0}_DRTYPE[1:0] = b00	request_type{3:0} = Single	
DMA{3:0}_DRTYPE[1:0] = b01	request_type{3:0} = Burst	
DMA{3:0}_DRLAST	The DMAC sets the state of the request_last flag:	
DMA{3:0}_DRLAST = 0	request_last{3:0} = 0	
DMA{3:0}_DRLAST = 1	request_last{3:0} = 1	

If the DMAC executes a DMAWFP single or DMAWFP burst instruction then the DMAC sets:

- The request_type{3:0} flag to Single or Burst, respectively
- The request_last{3:0} flag to 0

DMALPFE is an assembler directive which forces the associated DMALPEND instruction to have its nf bit set to 0. This creates a program loop that does not use a loop counter to terminate the loop. The DMAC exits the loop when the request_last flag is set to 1.

The DMAC conditionally executes the following instructions, depending on the state of the request_type and request_last flags:

DMALD, DMAST, DMALPEND

When these instructions use the optional B|S suffix then the DMAC executes a DMANOP if the request_type flag does not match.

DMALDP<B|S>, DMASTP<B|S>

The DMAC executes a DMANOP if the request_type flag does not match the B|S suffix.

DMALPEND

When the `nf` bit is 0, the DMAC executes a `DMANOP` if the `request_last` flag is set.

The `DMALDB`, `DMALDPB`, `DMASTB` and `DMASTPB` instructions should be used if the DMAC is required to issue a burst transfer when the DMAC receives a burst request, that is, when `DMA{3:0}_DRTYPE[1:0] = b01`. The values in the `CCRN` register control the amount of data that the DMAC transfers. See the Channel Control registers in [Appendix B, Register Details](#).

The `DMALDS`, `DMALDPS`, `DMASTS`, and `DMASTPS` instructions should be used if the DMAC is required to issue a single transfer when the DMAC receives a single request, that is, when `DMA{3:0}_DRTYPE[1:0] = b00`. The DMAC ignores the value of the `src_burst_len` and `dst_burst_len` fields in the `CCRN` register and sets the `arlen[3:0]` or `awlen[3:0]` buses to `0x0`.

See [Programming Examples Reference](#) for an example of microcode for peripheral length management.

DMAC Length Management

DMAC length management is the process by which the DMAC controls the total amount of data to transfer. Using the peripheral request interface, the peripheral notifies the DMAC when a transfer of data in either direction is required. The DMA channel thread controls how the DMAC responds to the peripheral requests.

The following constraints apply to DMAC length management:

- The total quantity of data for all of the single requests from a peripheral must be less than the quantity of data for a burst request for that peripheral.
- The `CCRN` register controls how much data is transferred for a burst request and a single request. ARM recommends that a `CCRN` register not be updated while a transfer is in progress for channel `n`. See the Channel Control registers in [Appendix B, Register Details](#).
- After the peripheral sends a burst request, the peripheral must not send a single request until the DMAC acknowledges that the burst request is complete.

The `DMAWFP` single instruction should be used when the program thread is required to halt execution until the peripheral request interface receives any request type. If the head entry request type in the request FIFO is:

Single: The DMAC pops the entry from the FIFO and continues program execution.

Burst: The DMAC leaves the entry in the FIFO and continues program execution.

Note: The burst request entry remains in the request FIFO until the DMAC executes a `DMAWFP` burst instruction or a `DMAFLUSHP` instruction.

The `DMAWFP` burst instruction should be used when the program thread is required to halt execution until the peripheral request interface receives a burst request. If the head entry request type in the request FIFO is:

Single: The DMAC removes the entry from the FIFO and program execution remains halted.

Burst: The DMAC pops the entry from the FIFO and continues program execution.

The DMAALDP instruction should be used when the DMAC is required to send an acknowledgement to the peripheral when it completes the AXI read transfers. Similarly, the DMASTP instruction should be used when the DMAC is required to send an acknowledgement to the peripheral when it completes the AXI write transfers. The DMAC uses the DMA{3:0}_DATYPE[1:0] bus to signal a transfer acknowledgement to the peripheral {3:0}.

The DMAC sends an acknowledgement for a read transaction when rvalid and rlast are High and for a write transaction when bvalid is High. If the system is able to buffer AXI write transfers, it might be possible for the DMAC to send an acknowledgement to the peripheral, but the transfer of write data to the end destination is still in progress.

The DMAFLUSHP instruction should be used to reset the request FIFO for the peripheral request interface. After the DMAC executes DMAFLUSHP, it ignores peripheral requests until the peripheral acknowledges the flush request. This enables the DMAC and peripheral to synchronize with each other.

See [Programming Examples Reference](#) for an example of microcode for DMA length management.

Peripheral Request Interface Timing Diagram

Figure 9-6 shows an example of the functional operation of the peripheral request interface using the rules that handshake rules described, when a peripheral requests a burst transfer.

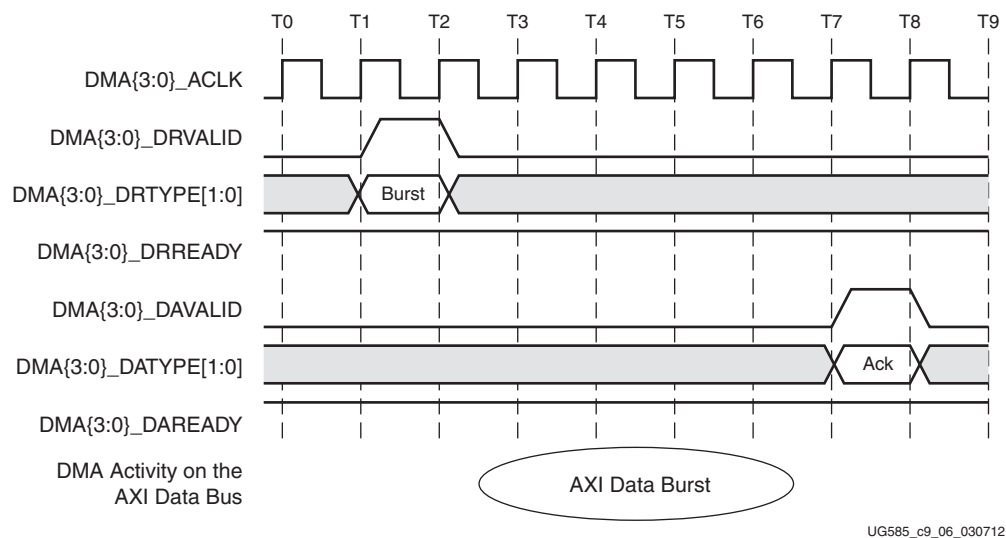


Figure 9-6: Burst Request Signaling

State transitions in Figure 9-6:

- | | |
|-------------------|--|
| T1 | The DMAC detects a request for a burst transfer. |
| Between T2 and T7 | The DMAC performs a burst transfer. |
| T7 | The DMAC sets DMA{3:0}_DAVALID High and sets DMA{3:0}_DATYPE[1:0] to indicate that the burst transfer is complete. |

For more timing diagrams refer to *ARM PrimeCell DMA Controller (PL330) Technical Reference Manual: Peripheral Request Interface Timing Diagrams*, keeping in mind that each peripheral request interface is asynchronous to one another and asynchronous to the DMA itself.

9.2.4 Multi-channel Data FIFO (MFIFO)

The MFIFO is a shared resource utilized on a first-come, first-served basis by all currently active channels. To a program, it appears as a set of variable-depth parallel FIFOs, one per channel, with the restriction that the total depth of all the FIFOs cannot exceed the size of the MFIFO. The DMAC maximum MFIFO depth is 128 64-bit words.

The controller is capable of realigning data from the source to the destination. For example, the DMAC shifts the data by two byte lanes when it reads a word from address `0x103` and writes to address `0x205`. The storage and packing of the data in the MFIFO is determined by the destination address and transfer characteristics.

When a program specifies that incrementing transactions are to be performed to the destination, the DMAC packs data into the MFIFO to minimize the usage of the MFIFO entries. For example, the DMAC packs two 32-bit words into a single entry in the MFIFO when the DMAC has a 64-bit AXI data bus and the program uses a source address of `0x100`, and destination address of `0x200`.

In certain situations, the number of entries required to store the data loaded from a source is not a simple calculation of the amount of source data divided by MFIFO width. The calculation of the number of entries required is not simple when any of the following occur:

- The source address is not aligned to the AXI bus width
- The destination address is not aligned to the AXI bus width
- The transactions are to a fixed destination, that is, a non-incrementing address

The DMALD and DMAST instructions each specify that an AXI transaction is to be performed. The amount of data transferred by an AXI transaction depends on the values programmed in to the CCRn register and the address of the transaction. See the *AMBA AXI Protocol Specification* for information about unaligned transfers.

See [Programming Examples Reference](#) for considerations about MFIFO utilization.

9.2.5 Events and Interrupts

The DMAC supports 16 event and interrupt signals.

Register `irq[7:0]` is used mainly as interrupts to the PS and the PL. These are parallel outputs going to both destinations at the same time. [Table 9-2](#) shows the mapping between DMAC IRQ# and System IRQ#. Each one of these event/interrupt resources can be programmed as either an event or an interrupt. See the Interrupt Enable register in [Appendix B, Register Details](#). Register `irq[15:8]` can be used only as events.

Table 9-2: Event/Interrupt Resources

DMAC IRQ#	System IRQ# (to the Processor)	System IRQ# (to the PL)	Event#
0 ~ 3	46 ~ 49	20 ~ 23	0 ~ 3
4 ~ 7	72 ~ 75	24 ~ 27	4 ~ 7
8 ~ 15			8 ~ 15

When the DMAC executes a DMASEV instruction it modifies the event/interrupt resource that the user specifies.

- If the INTEN register sets the event/interrupt resource to function as an event, the DMAC generates an event for the specified event/interrupt resource. When the DMAC executes a DMAWFE instruction for the same event-interrupt resource then it clears the event.
- If the INTEN register sets the event/interrupt resource to function as an interrupt, the DMAC sets irq<event_num> High, where event_num is the number of the specified event-resource. To clear the interrupt, the user must write to the INTCLR register. See the Interrupt Clear register in [Appendix B, Register Details](#).

See [Programming Examples Reference](#) for more information and [Chapter 7, Interrupts](#) for more details about the System IRQs.

9.2.6 Aborts

An abort is sent to the processors in the PS via IRQ#45 or to a PL peripheral via IRQ#28. [Table 9-3](#) summarizes all of the possible causes for an abort. [Table 9-4](#) explains the actions that the DMAC takes after an abort condition. After an abort occurs the action the DMAC takes depends on the thread type. [Table 9-5](#) describes the actions that the processors or the pl peripheral must take after the Abort signal is received. See *ARM PrimeCell DMA Controller (PL330) Technical Reference Manual: Aborts* for details.

Table 9-3: Abort Types and Conditions

Abort Types	Condition
<p>Precise</p> <p>The DMAC updates the PC register with the address of the instruction that created the abort.</p> <p>Note: When the DMAC signals a precise abort, the instruction that triggers the abort is not executed; the DMAC executes a DMANOP instead.</p>	<p>Security Violation on Channel Control Registers</p> <p>A DMA channel thread in a non-secure state attempts to program the Channel Control registers and generates a secure AXI transaction.</p>
	<p>Security Violation on Events</p> <p>A DMA channel thread in a non-secure state executes DMAWFE or DMASEV for an event that is set as secure. The SLCR register TZ_DMA_IRQ_NS controls the security state for an event.</p>
	<p>Security Violation on Peripheral Request Interfaces</p> <p>A DMA channel thread in a non-secure state executes DMAWFP, DMALDP, DMASTP, or DMAFLUSHP for a peripheral request interface that is set as secure. The SLCR register TZ_DMA_PERIPH_NS controls the security state for a peripheral request interface.</p>
	<p>Security Violation on DMAGO</p> <p>The DMA manager in a non-secure state executes DMAGO to attempt to start a secure DMA channel thread.</p>
	<p>Error on AXI Master Interface</p> <p>The DMAC receives an ERROR response on the AXI master interface when it performs an instruction fetch. For example; trying to access reserved memory.</p>
	<p>Error on Execution Engine</p> <p>A thread executes an undefined instruction or executes an instruction with an operand that is invalid for the configuration of the DMAC.</p>
<p>Imprecise</p> <p>The PC register might contain the address of an instruction that did not cause the abort occur.</p>	<p>Error on Data Load</p> <p>The DMAC receives an ERROR response on the AXI master interface when it performs a data load.</p>
	<p>Error on Data Store</p> <p>The DMAC receives an ERROR response on the AXI master interface when it performs a data store.</p>
	<p>Error on MFIFO</p> <p>A DMA channel thread executes DMALD and the MFIFO is too small to store the data or executes DMAST and the MFIFO contains insufficient data to complete the data transfer.</p>
	<p>Watchdog Abort</p> <p>The DMAC can lock up if one or more DMA channel programs are running and the MFIFO is too small to satisfy the storage requirements of the DMA programs. The DMAC contains logic to prevent it from remaining in a state where it is unable to complete a DMA transfer. The DMAC detects a lock up when all of the following conditions occur:</p> <ul style="list-style-type: none"> • Load queue is empty • Store queue is empty • All of the running channels are prevented from executing a DMALD instruction either because the MFIFO does not have sufficient free space or another channel owns the load-lock <p>When the DMAC detects a lock up it signals an interrupt and can also abort the contributing channels. The DMAC behavior depends on the state of the wd_irq_only bit in the WD register. For more information, see Resource Sharing Between DMA Channels.</p>

Table 9-4: Abort Handling

Thread Type	DMAC Actions
Channel thread	Sets IRQ#45 and IRQ#28 High
	Stops executing instructions for the DMA channel
	Invalidates all cache entries for the DMA channel
	Updates the Channel Program Counter registers to contain the address of the aborted instruction provided that the abort was precise
	Does not generate AXI accesses for any instructions remaining in the read queue and write queue
	Permits currently active AXI transactions to complete
DMA manager	Sets IRQ#45 and IRQ#28 High

Table 9-5: Thread Termination

Processor or PL Peripheral Actions
Reads the status of Fault Status DMA Manager register to determine if the DMA manager is faulting and to determine the cause of the abort
Reads the status of Fault Status DMA Channel register to determine if a DMA channel is faulting and to determine the cause of the abort
Programs the Debug Instruction-0 register with the encoding for the DMAKILL instruction
Writes to the Debug Command register.

9.2.7 Security Usage

When the DMAC exits from reset, the status of the reset initialization interface signals configures the security for the:

- DMA manager (SLCR register TZ_DMA_NS)
- Event/Interrupt resources (SLCR register TZ_DMA_IRQ_NS)
- Peripheral request interfaces (SLCR register TZ_DMA_PERIPH_NS)

See Section [Reset Initialization Interface](#) for more details.

When the DMA manager executes a DMAGO instruction for a DMA, it sets the security state of the channel setting the ns bit. See *ARM PrimeCell DMA Controller (PL330) Technical Reference Manual: DMAGO Instruction Encoding*. The status of the channel is provided by the dynamic non-secure bit, CNS in the Channel Status register.

[Table 9-6](#) describes how the nomenclature used in this chapter corresponds to ARM nomenclature. A quick summary of the security usage is given in [Table 9-7](#) for the DMA manager and in [Table 9-8](#) for the DMA channel threads. For more information refer to *ARM PrimeCell DMA Controller (PL330) Technical Reference Manual: Security Usage*.

Table 9-6: Security Usage Nomenclature

ARM Name	XILINX Name	Description
DNS	DMAC_NS in TZ_DMA_NS	Dma Non-secure When the DMAC exits from reset, this signal controls the security state of the DMA manager: 0: DMA manager operates in the secure state 1: DMA manager operates in the non-secure state
INS	DMAC_IRQ_NS<x> in TZ_DMA_IRQ_NS	Interrupt Non-secure When the DMAC exits from reset, this signal controls the security state of an event/interrupt: 0: DMAC interrupt/event bit is in the secure state 1: DMAC interrupt/event bit is in the non-secure state
PNS	DMAC_PERIPH_NS<x> in TZ_DMA_PERIPH_NS	Peripheral Non-secure When the DMAC exits from reset, this signal controls the security state of a peripheral request interface: 0: DMAC peripheral request interface is in the secure state 1: DMAC peripheral request interface is in the non-secure state
ns	ns in DMAGO instruction	DMAGO NON-SECURE Bit 1 of the DMAGO instruction: 0: DMA channel thread starts in the secure state 1: DMA channel thread starts in the secure state
CNS	CNS in CSR<x>	CHANNEL NON-SECURE The security state of each DMA channel is provided by bit CNS in the Channel Status register: 0: DMA channel thread operates in the secure state 1: DMA channel thread operates in the secure state

Table 9-7: Security Usage Summary for DMA Manager

	DNS	Instruction	ns	INS	Description
DMA Manager	0	DMAGO	0	-	The instruction must be issued using the secure APB interface. The DMA channel thread starts in secure state (CNS= 0).
			1	-	The instruction must be issued using the secure APB interface. The DMA channel thread starts in non-secure state (CNS=1).
		DMASEV	-	X	The instruction must be issued using the secure APB interface. It signals the appropriate event irrespective of the INS bit.
	1	DMAGO	0	-	The instruction must be issued using the non-secure APB interface. Abort
			1	-	The instruction must be issued using the non-secure APB interface. The DMA channel thread starts in non-secure state (CNS=1).
		DMASEV	-	0	The instruction must be issued using the non-secure APB interface. Abort
			-	1	The instruction must be issued using the non-secure APB interface. It signals the appropriate event.

Table 9-8: Security Usage Summary for DMA Channel Thread

	CNS	Instruction	PNS	INS	Description
DMA Channel Thread	0	DMAWFE	-	X	On event, execution continues, irrespective of the INS bit
		DMASEV	-	X	Signals the appropriate event, irrespective of the INS bit
		DMAWFP	X	-	On peripheral request, execution continues, irrespective of the PNS bit
		DMALP, DMASTP	X	-	Sends a message to the peripheral to communicate that data transfer is complete, irrespective of the PNS bit
		DMAFLUSH	X	-	Clears the state of the peripheral and sends a message to the peripheral to resend its level status, irrespective of the PNS bit
	1	DMAWFE	-	0	Abort
			-	1	On event, execution continues
		DMASEV	-	0	Abort
			-	1	It signals the appropriate event
		DMAWFP	0	-	Abort
			1	-	On peripheral request, execution continues
		DMALP, DMASTP	0	-	Abort
			1	-	Sends a message to the peripheral to communicate that data transfer is complete
		DMAFLUSHP	0	-	Abort
			1	-	It only clears the state of the peripheral and sends a message to the peripheral to resend its level status

9.3 External Signals

9.3.1 Peripheral Request Interface

The peripheral request interfaces support the connection of DMA-capable peripherals to enable memory-to-peripheral and peripheral-to-memory DMA transfers to occur, without intervention from a microprocessor. These peripherals must be in the PL and attached to the M_AXI_GP interface. All peripheral request interface signals are synchronous to the respective clocks.

Table 9-9: PL Peripheral Request Interface Signals

Type	I/O	Name	Description
Clock and Reset	I	DMA{3:0}_ACLK	Clock for DMA request transfers
	O	DMA{3:0}_RSTN	Reset DMA interface

Table 9-9: PL Peripheral Request Interface Signals (Cont'd)

Type	I/O	Name	Description
DMA Request	I	DMA{3:0}_DRVALID	Indicates when the peripheral provides valid control information: 0: No control information is available 1: DMA{3:0}_DRTYPE[1:0] and DMA{3:0}_DRLAST contain valid information for the DMAC
	I	DMA{3:0}_DRLAST	Indicates that the peripheral is sending the last data transfer for the current DMA transfer: 0: Last data request is not in progress 1: Last data request is in progress Note: The DMAC only uses this signal when DMA{3:0}_DRTYPE[1:0] is b00 or b01.
	I	DMA{3:0}_DRTYPE[1:0]	Indicates the type of acknowledgement, or request, that the peripheral signals: 00: Single level request 01: Burst level request 10: Acknowledging a flush request that the DMAC requested 11: Reserved
	O	DMA{3:0}_DRREADY	Indicates if the DMAC can accept the information that the peripheral provides on DMA{3:0}_DRTYPE[1:0]: 0: DMAC not ready 1: DMAC ready
DMA Acknowledge	O	DMA{3:0}_DAVALID	Indicates when the DMAC provides valid control information: 0: No control information is available 1: DMA{3:0}_DATYPE[1:0] contains valid information for the peripheral
	I	DMA{3:0}_DAREADY	Indicates if the peripheral can accept the information that the DMAC provides on DMA{3:0}_DATYPE[1:0]: 0: Peripheral not ready 1: Peripheral ready
	I	DMA{3:0}_DATYPE[1:0]	Indicates the type of acknowledgement, or request, that the DMAC signals: 00: DMAC has completed the single DMA transfer 01: DMAC has completed the burst DMA transfer 10: DMAC requesting the peripheral to perform a flush request 11: Reserved

9.3.2 AXI Master Interface

The DMAC contains an AXI master interface that enables it to transfer data from a source AXI slave to a destination AXI slave.

See *ARM PrimeCell DMA Controller (PL330) Technical Reference Manual: AXI characteristics for a DMA transfer and AXI master interface* for more information.

9.3.3 Reset Initialization Interface

Table 9-10 shows the tie-off signals used to program security state of the DMAC. Depending on the state of the SLCR registers after reset, the DMA is configured in secure or non-secure mode. See *ARM PrimeCell DMA Controller (PL330) Technical Reference Manual: Security Usage* for more details.

Table 9-10: DMAC Initialization Signals

Name	Type	Source	Description
boot_manager_ns	Input	SLCR register TZ_DMA_NS	Controls the security state of the DMA manager, when the DMAC exits from reset: 0: Assigns DMA manager to the secure state 1: Assigns DMA manager to the non-secure state
boot_irq_ns[15:0]	Input	SLCR register TZ_DMA_IRQ_NS	Controls the security state of an event-interrupt resource, when the DMAC exits from reset: <ul style="list-style-type: none"> boot_irq_ns[x] is Low: Assigns event<x> or irq[x] to the secure state boot_irq_ns[x] is High: Assigns event<x> or irq[x] to the non-secure state
boot_periph_ns[3:0]	Input	SLCR register TZ_DMA_PERIPH_NS	Controls the security state of a peripheral request interface, when the DMAC exits from reset: <ul style="list-style-type: none"> boot_periph_ns[x] is Low: Assigns peripheral request interface x to the secure state boot_periph_ns[x] is High: Assigns peripheral request interface x to the non-secure state
boot_addr[31:0]	Input	Hard-wired 32'h0	Configures the address location that contains the first instruction that the DMAC executes, when the DMAC exits from reset. Note: The DMAC only uses this address when boot_from_pc is High.
boot_from_pc	Input	Hard-wired 1'b0	Controls the location of where the DMAC executes its initial instruction, after the DMAC exits from reset: 0: DMAC waits for an instruction from either APB interface 1: DMA manager executes the instruction that is located at the address provided by boot_addr[31:0]

9.4 Register Overview

Table 9-11 provides an overview of the DMAC registers.

Table 9-11: DMA Controller Register Overview

Function	Register Names	Overview
Main Control	XDMAPS_DS XDMAPS_DPC	Provides the security state and the program counter.
Interrupt Processing	INT_EVENT_RIS INTCLR INTEN INTMIS	Enables/disables the interrupt detection, mask interrupt sent to the interrupt controller, and reads raw interrupt status.
Fault Status and Type	FSRD FSRC FTRD FTR{7:0}	Provides the fault status and type for the manager and the channels.
Channel Status	CPC{7:0} CSR{7:0} SAR{7:0} DAR{7:0} CCR{7:0} LC0_{7:0} LC1_{7:0}	These registers provide the status of the DMA channel threads.
Debug	DBGSTATUS DBGCMD DBGINST{1,0}	These registers enable the user to send instructions to a channel thread.
Configuration	XDMAPS_CR{4:0} XDMAPS_CRDN	These registers enable system firmware to discover the configuration of the DMAC and control the behavior of the watchdog.
Watchdog	WD	Controls how the DMAC responds when it detects a lock-up condition.
System-level	DMAC_RST_CTRL FPGA_RST_CTRL[11:8] TZ_DMACH_NS TZ_DMA_IRQ_NS TZ_DMACH_PERIPH_NS DMACH_RAM APER_CLK_CTRL	Control reset, clock, and security state.

9.5 Instruction Set Reference for Manager and Commands

Table 9-12 and Table 9-13 summarize the DMAC instructions and commands.

Table 9-12: DMA Controller Instruction Summary

Instruction	Mnemonic	Thread Usage: M = DMA Manager C = DMA Channel	
Add Halfword	DMAADDH	-	C
Add Negative Halfword	DMAADNH	-	C
End	DMAEND	-	C
Flush and Notify Peripheral	DMAFLUSHP	-	C
Go	DMAGO	M	-
Kill	DMAKILL	M	C
Load	DMALD	-	C
Load and Notify Peripheral	DMALDP	-	C
Loop	DMAALP	-	C
Loop End	DMAALPEND	-	C
Loop Forever	DMAALPFE	-	C
Move	DMAMOV	-	C
No operation	DMANOP	M	C
Read memory Barrier	DMARMB	-	C
Send Event	DMASEV	M	C
Store	DMAST	-	C
Store and Notify Peripheral	DMASTP	-	C
Store Zero	DMASEV	-	C
Wait For Event	DMAWFE	-	C
Wait For Peripheral	DMAWFP	-	C
Write memory Barrier	DMAWMB	-	C

Table 9-13: Additional Commands Provided by the Assembler

Directives	Mnemonic
Place a 32-bit immediate	DCD
Place a 8-bit immediate	DCB
Loop	DMALP
Loop Forever	DMALPFE

Table 9-13: Additional Commands Provided by the Assembler (Cont'd)

Directives	Mnemonic
Loop End	DMALPEND
Move CCR	DMAMOVCCR

See *ARM PrimeCell DMA Controller (PL330) Technical Reference Manual: Instruction Set* for more information about instructions and assemble directives.

9.6 Programming Examples Reference

(See *ARM Application Note 239: Example programs for the CoreLink DMA Controller DMA-330* for more programming examples.)

9.6.1 Start a DMA Channel Thread

The following example shows the steps required to start a DMA channel thread using the debug instruction registers.

1. Create a program for the DMA channel.
2. Store the program in a region of system memory.

Use one of the APB interfaces on the DMAC to program a DMAGO instruction as follows:

3. Poll the DBGSTATUS register to ensure that debug is idle, that is, the dbgstatus bit is 0. See the Debug Status register in [Appendix B, Register Details](#).
4. Write to the DBGINST0 register and enter the:
 - a. Instruction byte 0 encoding for DMAGO.
 - b. Instruction byte 1 encoding for DMAGO.
 - c. Debug thread bit to 0. This selects the DMA manager. See the Debug Instruction-0 register in [Appendix B, Register Details](#).
5. Write to the DBGINST1 register with the DMAGO instruction byte [5:2] data, see the Debug Instruction-1 register in [Appendix B, Register Details](#). These four bytes must be set to the address of the first instruction in the program that was written to system memory in Step 2.

Instruct the DMAC to execute the instruction that the debug instruction registers contain:

6. Write a 0 to the DBGCMD register. The DMAC starts the DMA channel thread and sets the dbgstatus bit to 1. See the Debug Command register in [Appendix B, Register Details](#). After the DMAC completes execution of the instruction, it clears the dbgstatus bit to 0.

9.6.2 Memory to Memory

This section shows examples of microcode that the DMAC executes to perform aligned, unaligned, and fixed transfers. See [Table 9-12](#) for aligned transfer, [Table 9-12](#) for unaligned transfer, and [Table 9-12](#) for Fixed transfer. MFIFO utilization is also described.

Table 9-14: Aligned Transfers

Description	Code	MFIFO Usage
Simple Aligned Program In this program the source address and destination address are aligned with the AXI data bus width.	DMAMOV CCR, SB4 SS64 DB4 DS64 DMAMOV SAR, 0x1000 DMAMOV DAR, 0x4000 DMALP 16 DMALD DMAST DMALPEND DMAEND	Each DMALD requires four entries and each DMAST removes four entries. This example has a static requirement of zero MFIFO entries and a dynamic requirement of four MFIFO entries.
Aligned asymmetric program with multiple loads The following program performs four loads for each store and the source address and destination address are aligned with the AXI data bus width.	DMAMOV CCR, SB1 SS64 DB4 DS64 DMAMOV SAR, 0x1000 DMAMOV DAR, 0x4000 DMALP 16 DMALD DMALD DMALD DMALD DMAST DMALPEND	Each DMALD requires four entries and each DMAST removes one entry. This example has a static requirement of zero MFIFO entries and a dynamic requirement of four MFIFO entries.
Aligned asymmetric program with multiple stores In this program, the source address is aligned with the AXI data bus width but the destination address is unaligned. The destination address is not aligned to the destination burst size so the first DMAST instruction removes less data than the first DMALD instruction reads. Therefore, a final DMAST of a single word is required to clear the data from the MFIFO.	DMAMOV CCR, SB4 SS64 DB1 DS64 DMAMOV SAR, 0x1000 DMAMOV DAR, 0x4000 DMALP 16 DMALD DMAST DMAST DMAST DMAST DMALPEND DMAEND	Each DMALD requires four entries and each DMAST removes one entry. This example has a static requirement of zero MFIFO entries and a dynamic requirement of four MFIFO entries.

Table 9-15: Unaligned Transfers

Description	Code	MFIFO Usage
Aligned source address to unaligned destination address In this program, the source address is aligned with the AXI data bus width but the destination address is unaligned. The destination address is not aligned to the destination burst size so the first DMAST instruction removes less data than the first DMALD instruction reads. Therefore, a final DMAST of a single word is required to clear the data from the MFIFO.	DMAMOV CCR, SB4 SS64 DB4 DS64 DMAMOV SAR, 0x1000 DMAMOV DAR, 0x4004 DMALP 16 DMALD DMAST DMALPEND DMAMOV CCR, SB4 SS64 DB1 DS32 DMAST DMAEND	The first DMALD instruction loads four double words but because the destination address is unaligned, the DMAC shifts them by four bytes and therefore it uses five entries in the MFIFO. Each DMAST requires only four entries of data and therefore the extra entry remains in use for the duration of the program until it is emptied by the last DMAST. This example has a static requirement of one MFIFO entry and a dynamic requirement of four MFIFO entries.
Unaligned source address to aligned destination address In this program the source address is unaligned with the AXI data bus width but the destination address is aligned. The source address is not aligned to the source burst size so the first DMALD instruction reads in less data than the DMAST requires. Therefore, an extra DMALD is required to satisfy the first DMAST.	DMAMOV CCR, SB4 SS64 DB4 DS64 DMAMOV SAR, 0x1004 DMAMOV DAR, 0x4000 DMALD DMALP 15 DMALD DMAST DMALPEND DMAMOV CCR, SB1 SS32 DB4 DS64 DMALD DMAST DMAEND	The first DMALD instruction loads five beats of data to enable the DMAC to execute the first DMAST. After the first DMALD, the subsequent DMALDs are not aligned to the source burst size, for example the second DMALD reads from address 0x1028. After the loop, the final two DMALDs read the data required to satisfy the final DMAST. This example has a static requirement of one MFIFO entry and a dynamic requirement of four MFIFO entries.

Table 9-15: Unaligned Transfers (Cont'd)

Description	Code	MFIFO Usage
<p>Unaligned source address to aligned destination address, with excess initial load</p> <p>This program is an alternative to that described in unaligned source address to aligned destination address. The program uses a different sequence of source bursts which might be less efficient but requires fewer MFIFO entries.</p>	<pre> DMAMOV CCR, SB5 SS64 DB4 DS64 DMAMOV SAR, 0x1004 DMAMOV DAR, 0x4000 DMALD DMAST DMAMOV CCR, SB4 SS64 DB4 DS64 DMALP 14 DMALD DMAST DMALPEND DMAMOV CCR, SB3 SS64 DB4 DS64 DMALD DMAMOV CCR, SB1 SS32 DB4 DS64 DMALD DMAST DMAEND </pre>	<p>The first DMALD instruction loads five beats of data to enable the DMAC to execute the first DMAST. After the first DMALD, the subsequent DMALDs are not aligned to the source burst size, for example the second DMALD reads from address 0x1028. After the loop, the final two DMALDs read the data required to satisfy the final DMAST. This example has a static requirement of one MFIFO entry and a dynamic requirement of four MFIFO entries.</p>
<p>Aligned burst size, unaligned MFIFO</p> <p>In this program the destination address, which is narrower than the MFIFO width, aligns with the burst size but does not align with the MFIFO width.</p>	<pre> DMAMOV CCR, SB4 SS32 DB4 DS32 DMAMOV SAR, 0x1000 DMAMOV DAR, 0x4004 DMALP 16 DMALD DMAST DMALPEND DMAEND </pre>	<p>If the DMAC configuration has a 32-bit AXI data bus width then this program requires four MFIFO entries. However, in this example the DMAC has a 64-bit AXI data bus width and, because the destination address is not 64-bit aligned, it requires three rather than the expected two MFIFO entries. This example has a static requirement of zero MFIFO entries and a dynamic requirement of three MFIFO entries.</p>

Table 9-16: Fixed Transfers

Description	Code	MFIFO Usage
<p>Fixed destination with aligned address</p> <p>In this program the source address and destination address are aligned with the AXI data bus width, and the destination address is fixed.</p>	<pre> DMAMOV CCR, SB2 SS64 DB4 DS32 DAF DMAMOV SAR, 0x1000 DMAMOV DAR, 0x4000 DMALP 16 DMALD DMAST DMALPEND DMAEND </pre>	<p>Each DMALD in the program loads two 64-bit data transfers into the MFIFO. Because the destination address is a 32-bit fixed address then the DMAC splits each 64-bit data item across two entries in the MFIFO. This example has a static requirement of zero MFIFO entries and a dynamic requirement of four MFIFO entries.</p>

9.6.3 Memory to/from PL Peripheral

There are two different way to handle the quantity of the data flowing between the DMAC and the peripheral:

- Peripheral length management.
 - The peripheral controls the quantity of data that a DMA cycle contains, without the DMAC being aware of how many data transfer that it contains.
- DMAC length management.
 - The DMAC is controlling the quantity of data in a DMA cycle.

Example Program for Peripheral Length Management

The following example shows a DMAC program that transfers 64 words from memory to peripheral zero when the peripheral sends a burst request (DMA{3:0}_DRTYPE[1:0] = b01). When the peripheral sends a single request (DMA{3:0}_DRTYPE[1:0] = b00) then the DMAC program transfers one word from memory to peripheral zero.

To transfer the 64 words, the program instructs the DMAC to perform 16 AXI transfers. Each AXI transfer consists of a 4-beat burst (SB=4, DB=4), each beat of which moves a word of data (SS=32, DS=32).

In this example, the program shows use of the following instructions:

- DMAWFP instruction. The DMAC waits for either a burst or single request from the peripheral.
- DMASTPB and DMASTPS instructions. The DMAC informs the peripheral when a transfer is complete.

```
# Set up for burst transfers (4-beat burst, so SB4 and DB4),
# (word data width, so SS32 and DS32)
DMAMOV CCR SB4 SS32 DB4 DS32
DMAMOV SAR ...
DMAMOV DAR ...

# Initialize peripheral '0'
DMAFLUSHP P0

# Perform peripheral transfers
# Outer loop - DMAC responds to peripheral requests until peripheral
# sets drlast_0 = 1
DMALPFE

# Wait for request, DMAC sets request_type0 flag depending on the
# request type it receives
DMAWFP 0, periph

# Set up loop for burst request: first 15 of 16 sets of transactions
# Note: B suffix - conditionally executed only if request_type0
# flag = Burst
DMALP 15
  DMALDB
  DMASTB
```



```

# Only loopback if servicing a burst, otherwise treat as a NOP
DMALPENDB

# Perform final transaction (16 of 16). Send the peripheral
# acknowledgement of burst request completion
DMALDB
DMASTPB P0

# Perform transaction if the peripheral signals a single request
# Note: S suffix - conditionally executed only if request_type0
# flag = Single
DMALDS
DMASTPS P0

# Exit loop if DMAC receives the last request, that is, drlast_0 = 1
DMALPEND

DMAEND

```

Example Program for DMAC Length Management

This example shows a DMAC program that can transfer 1,027 words when a peripheral signals 16 consecutive burst requests and 3 consecutive single requests.

```

# Set up for AXI burst transfer
# (4-beat burst, so SB4 and DB4), (word data width, so SS32 and DS32)
DMAMOV CCR SB4 SS32 DB4 DS32
DMAMOV SAR ...
DMAMOV DAR ...

# Initialize peripheral '0'
DMAFLUSHP P0

# Perform peripheral transfers
# Burst request loop to transfer 1024 words
DMALP 16

# Wait for the peripheral to signal a burst request.
# DMAC transfers 64 words for each burst request
DMAWFP 0, burst

# Set up loop for burst request: first 15 of 16 sets of transactions
DMALP 15
    DMALD
    DMAST
DMALPEND

# Perform final transaction (16 of 16).
# Send the peripheral acknowledgement of burst request completion
DMALD
DMASTPB 0

# Finish burst loop
DMALPEND

# Set up for AXI single transfer (word data width, so SS32 and DS32)
DMAMOV CCR SB1 SS32 DB1 DS32

```

```
# Single request loop to transfer 3 words
DMALP 3

# Wait for the peripheral to signal a single request. DMAC to transfer
# one word
DMAWFP 0, single

# Perform transaction for single request and send completion
# acknowledgement to the peripheral
DMALDS
DMASTPS P0

# Finish single loop
DMALPEND

# Flush the peripheral, in case the single transfers were in response
# to a burst request
DMAFLUSHP 0

DMAEND
```

9.6.4 Using an Event to Restart DMA Channels

When the INTEN register is programmed to generate an event, the DMASEV and DMAWFE instructions can be used to restart one or more DMA channels. See the Interrupt Enable register in [Appendix B, Register Details](#).

The following sections describe the DMAC behavior when:

- DMAC executes DMAWFE before DMASEV
- DMAC executes DMASEV before DMAWFE

DMAC Executes DMAWFE before DMASEV

To restart a single DMA channel:

1. The first DMA channel executes DMAWFE and then stalls while it waits for the event to occur.
2. The other DMA channel executes DMASEV using the same event number. This generates an event, and the first DMA channel restarts. The DMAC clears the event, one DMA{3:0}_ACLK cycle after it executes DMASEV.

Multiple channels can be programmed to wait for the same event. For example, if four DMA channels have all executed DMAWFE for event 12, then when another DMA channel executes DMASEV for event 12, the four DMA channels all restart at the same time. The DMAC clears the event one clock cycle after it executes DMASEV.

DMAC Executes DMASEV before DMAWFE

If the DMAC executes DMASEV before another channel executes DMAWFE, then the event remains pending until the DMAC executes DMAWFE. When the DMAC executes DMAWFE, it halts execution for one DMA{3:0}_ACLK cycle, clears the event, and then continues execution of the channel thread.

For example, if the DMAC executes DMASEV 6 and none of the other threads have executed DMAWFE 6, then the event remains pending. If the DMAC executes DMAWFE 6 instruction for channel 4 and then executes DMAWFE 6 instruction for channel 3, then:

1. The DMAC halts execution of the channel 4 thread for one DMA{3:0}_ACLK cycle.
2. The DMAC clears event 6.
3. The DMAC resumes execution of the channel 4 thread.
4. The DMAC halts execution of the channel cycle after it executes DMASEV.

9.6.5 Interrupting a Processor

The DMAC provides the irq[7:0] signals for use as active-High level-sensitive interrupts to external microprocessors. When the INTEN register is programmed to generate an interrupt, after the DMAC executes DMASEV, it sets the corresponding irq[7:0] High. See Interrupt Enable register in [Appendix B, Register Details](#).

An external microprocessor can clear the interrupt by writing to the Interrupt Clear register.

Executing DMAWFE does not clear an interrupt.

If the DMASEV instruction is used to notify a microprocessor when the DMAC completes a DMALD or DMAST instruction, ARM recommends that a memory barrier instruction be inserted before the DMASEV. Otherwise the DMAC might signal an interrupt before the AXI transfers complete.

This is demonstrated in the following example:

```

DMALD
DMAST

# Issue a write memory barrier
# Wait for the AXI write transfer to complete before the DMAC can
# send an interrupt
DMAWMB

# The DMAC sends the interrupt
DMASEV

```

9.7 Programming Restrictions

See *ARM PrimeCell DMA Controller (PL330) Technical Reference Manual: Programming Restrictions* for details about restrictions that apply when programming the DMAC:

- Fixed unaligned bursts
- Endian swap size restrictions
- Updating DMA channel control registers during a DMA cycle
- Full MFIFO causes DMAC watchdog to abort a DMA channel. (Resource sharing between DMA channels)

The following sections describe two of these restrictions in detail.

Updating DMA Channel Control Registers During a DMA Cycle

Prior to the DMAC executing a sequence of DMALD and DMAST instructions, the values by the user in to the CCRn register, SARn register, and DARn register control the data byte lane manipulation that the DMAC performs when it transfers the data from the source address to the destination address. See the Channel Control registers, Source Address registers, and Destination Address registers in [Appendix B, Register Details](#).

These registers can be updated during a DMA cycle, but if certain register fields are changed, the DMAC might discard data. The following sections describe the register fields that might have a detrimental impact on a data transfer:

- Updates that affect the destination address
- Updates that affect the source address

Updates That Affect the Destination Address

If a DMAMOV instruction is used to update the DARn register or CCRn register part way through a DMA cycle, a discontinuity in the destination datastream might occur. A discontinuity occurs if any of the following is changed:

- `dst_inc` bit
- `dst_burst_size` field when `dst_inc` = 0, (fixed-address burst)
- DARn register so that it modifies the destination byte lane alignment. For example, when the bus width is 64 bits and bits [2:0] in the DARn register are changed.

When a discontinuity in the destination datastream occurs, the DMAC:

1. Halts execution of the DMA channel thread.
2. Completes all outstanding read and write operations for the channel (just as if the DMAC was executing DMARMB and DMAWMB instructions).
3. Discards any residual MFIFO data for the channel.
4. Resumes execution of the DMA channel thread.

Updates That Affect the Source Address

If a DMAMOV instruction is used to update the SARn register or CCRn register part way through a DMA cycle, a discontinuity in the source datastream might occur. A discontinuity occurs if any of the following is changed:

- `src_inc` bit
- `src_burst_size` field
- SARn register so that it modifies the source byte lane alignment. For example, when the bus width is 32 bits and bits [1:0] in the SARn register are changed.

When a discontinuity in the source datastream occurs, the DMAC:

1. Halts execution of the DMA channel thread.
2. Completes all outstanding read operations for the channel (just as if the DMAC was executing DMARMB instruction).
3. Resumes execution of the DMA channel thread. No data is discarded from the MFIFO.

Resource Sharing Between DMA Channels

DMA channel programs share the MFIFO data storage resource. A set of concurrently running DMA channel programs must not be started with a resource requirement that exceeds the configured size of the MFIFO. If this limit is exceeded, the DMAC might lock up and generate a watchdog abort.

The DMAC includes a mechanism called the load-lock to ensure that the shared MFIFO resource is used correctly. The load-lock is either owned by one channel, or it is free. The channel that owns the load-lock can execute DMALD instructions successfully. A channel that does not own the load-lock pauses at a DMALD instruction until it takes ownership of the load-lock.

A channel claims ownership of the load lock when:

- It executes a DMALD or DMALDP instruction.
- No other channel currently owns the load-lock.

A channel releases ownership of the load-lock when any of the following occur:

- It executes a DMAST, DMASTP, or DMASTZ.
- It reaches a barrier, that is, it executes DMARMB or DMAWMB.
- It waits, that is, it executes DMAWFP or DMAWFE.
- It terminates normally, that is, it executes DMAEND.
- It aborts for any reason, including DMAKILL.

The MFIFO resource usage of a DMA channel program is measured in MFIFO entries, and rises and falls as the program proceeds. The MFIFO resource requirement of a DMA channel program is described using a static requirement and a dynamic requirement which are affected by the load-lock mechanism.

ARM defines the static requirement to be the maximum number of MFIFO entries that a channel is currently using before that channel does one of the following:

- Executes a WFP or WFE instruction.
- Claims ownership of the load-lock.

ARM defines the dynamic requirement to be the difference between the static requirement and the maximum number of MFIFO entries that a channel program uses at any time during its execution.

To calculate the total MFIFO requirement, add the largest dynamic requirement to the sum of all the static requirements.

To avoid DMAC lock-up, the total MFIFO requirement of the set of channel programs must be equal to or less than the maximum MFIFO depth. The DMAC maximum MFIFO depth is 128 64-bit words. See [Multi-channel Data FIFO \(MFIFO\)](#) for more information.

9.8 DMAC IP Configuration Options

The Xilinx implementation of the DMAC uses the IP configuration options shown in [Table 9-17](#).

Table 9-17: DMAC IP Configuration Options

IP Configuration Option	Value
Data width (bits)	64
Number of channels	8
Number of interrupts	16 (8 interrupts, 8 events)
Number of peripherals	4 (to programmable logic)
Number of cache lines	8
Cache line width (words)	4
Buffer depth (MFIFO depth)	128
Read queue depth	16
Write queue depth	16
Read issuing capability	8
Write issuing capability	8
Peripheral request capabilities	All capabilities
Secure APB base address	0xF800_3000
Non-secure APB base address	0xF800_4000

DDR Memory Controller

10.1 Introduction

The DDR memory controller supports DDR2, DDR3, and LPDDR2 devices and consists of three major blocks: an AXI memory port interface (DDRI), a core controller with transaction scheduler (DDRC) and a controller with digital PHY (DDRP).

The AXI interfaces with four 64-bit synchronous AXI interfaces, to serve multiple AXI masters simultaneously. Each AXI interface has its own dedicated transaction FIFO.

The DDRC contains two 32-entry content addressable memories (CAMs) to perform DDR data service scheduling to maximize DDR memory efficiency. It also contains fly-by channel for low latency channel to allow access to DDR memory without going through the CAM.

The PHY processes read/write requests from the controller and translates them into specific signals within the timing constraints of the target DDR memory. Signals from the controller are used by the PHY to produce internal signals that connect to the pins via the digital PHYs. The DDR pins connect directly to the DDR device(s) via the PCB signal traces.

The system accesses the DDR via DDRI via its four 64-bit AXI memory ports. One AXI port is dedicated to the L2-cache for the CPUs and ACP, two ports are dedicated to the AXI_HP interfaces, and the fourth port is shared by all the other masters on the AXI interconnect.

The DDR interface (DDRI) arbitrates the requests from the eight ports (four reads and four writes). The arbiter selects a request and passes it to the DDR controller and transaction scheduler (DDRC). The arbitration is based on a combination of how long the request has been waiting, the urgency of the request, and if the request is within the same page as the previous request.

The DDRC receives requests from the DDRI through a single interface. Both reads and writes flow through this interface. Read requests include a tag field that is returned with the data from the DDR. The DDR controller PHY (DDRP) drives the DDR transactions.

10.1.1 Features

DDR Controller PHY

The DDR controller PHY has these features:

- DDR memories

- 1.2V LPDDR2
- 1.8V DDR2
- 1.5V DDR3
- Selectable 16-bit and 32-bit data bus widths
- Optional ECC in 16-bit data width configuration
- Self-refresh entry on software command and automatic exit on command arrival
- Autonomous DDR power down entry and exit based on programmable idle periods
- Data read strobe auto-calibration
- AXI memory interface ports
- Four 64-Bit AXI interfaces with separate Read/Write ports and 32-bit addressing
- Write data byte enables supported for each data beat
- Low latency read mechanism using HPR queue
- Special urgent signaling to each port
- TrustZone regions programmable on 64 MB boundaries
- Exclusive accesses for two different IDs per port (locked transactions are not supported)

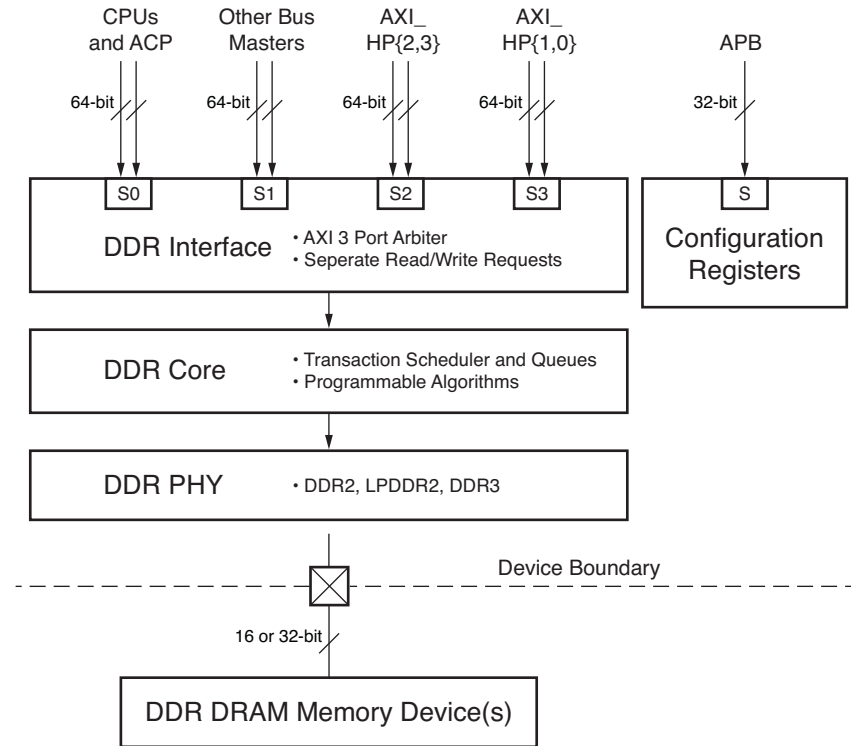
DDR Controller Core and Transaction Scheduler

The DDR controller core and transaction scheduler has these features:

- Transaction scheduling is done to optimize data bandwidth and latency
- Advanced re-ordering engine to maximize memory access efficiency for continuous reads and writes as well as random reads and writes
- Write-read address collision checking that flushes the write buffer
- Obeys AXI ordering rules

10.1.2 Block Diagram

The block diagram for the DDR memory controller is shown in [Figure 10-1](#). The DDR memory controller consists of an arbiter, a core with transaction scheduler, and the physical sequencing of the DDR memory signals.



UG585_c10_01_032012

Figure 10-1: DDR Memory Controller Block Diagram

The controller core and transaction scheduler contains two 32-entry CAMs to perform DDR data service re-ordering to maximize DDR memory access efficiency. It also contains fly-by channel for low latency channel to allow access to DDR memory without going through the CAM.

The PHY processes read/write requests from the controller and translates them into specific signals within the timing constraints of the target DDR memory. Signals from the controller are used by the PHY to produce internal signals that connect to the pads of the PS using the PHY. The pads connect directly, via the PCB signal traces, to the external memory devices.

10.1.3 Notices

7z010 CLG225 Device Note

All devices support the 32- and 16-bit data bus width options except the 7z010 CLG225 device. The 7z010 supports only the 16-bit data bus width, not the 32-bit bus.

10.1.4 Interconnect

The four AXI_HP interfaces are multiplexed down, in pairs, and are connected to ports 2 and 3 as shown in Figure 10-2. These ports are commonly configured for high bandwidth traffic. The path from these four interfaces to the DDR include two ports on the DDR memory port arbiter. The interconnect switch arbitrates back-and-forth between each of the two ports. Read and write

channels operate separately. The arbitration in the bridge can be affected by the QoS signals from each PL interface. A requestor with a higher QoS value is given preferential treatment by the interconnect bridge. Arbitration is priority based using QoS as priority. In the event of a tie, an LRG scheme is used to break the tie.

The L2-cache is connected to port 0 and is used to serve the CPUs and the ACP interface to the PL. This port is commonly configured for low-latency. The other masters on the AXI interconnect are connected to port 1.

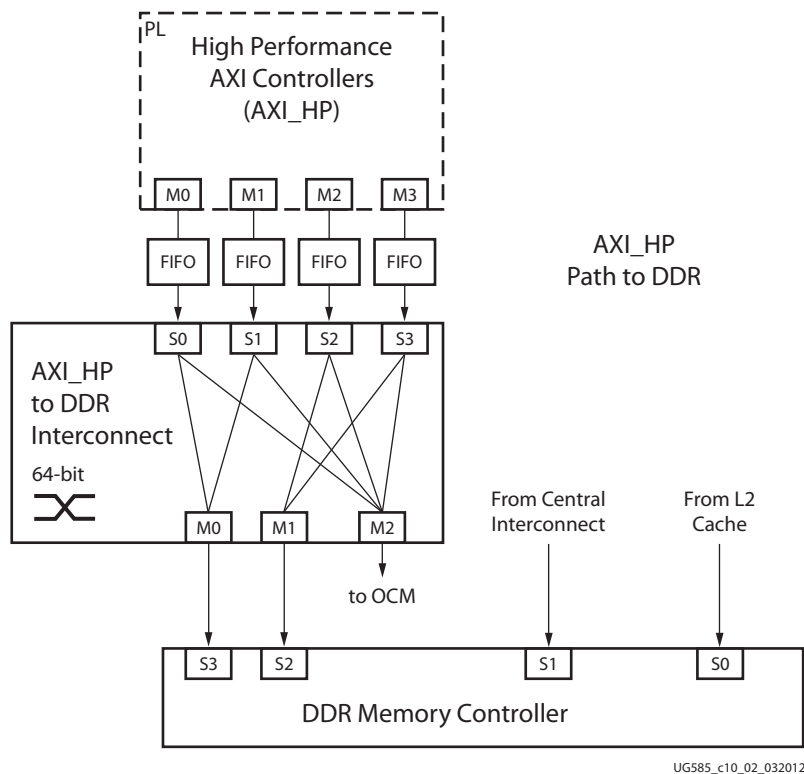


Figure 10-2: DDRC System Viewpoint

10.1.5 DDR Memory Types, Densities, and Data Widths

The DDR memory controller is able to connect to devices under the conditions identified in Table 10-1.

Table 10-1: Connectivity Limitations

Parameter	Value	Notes
Maximum Total Memory Density	1 GB	1 GB of address map is allocated to DRAM
Total Data Width (bits)	16, 32	ECC can only use a 32-bit configuration: 16 data bits, 10 check bits
Component Data Width (bits)	8, 16, 32	4-bit devices are not supported
Maximum Ranks	1	

Table 10-1: Connectivity Limitations (Cont'd)

Parameter	Value	Notes
Maximum Row Address (bits)	15	
Maximum Bank Address (bits)	3	

Table 10-2 provides a collection of example memory configurations.

Table 10-2: Example Memory Configurations

Technology	Component Configuration	Number of Components	Component Density	Total Width	Total Density
DDR3	x16	2	4 Gb	32	1 GB
DDR2	x8	4	2 Gb	32	1 GB
LPDDR2	x32	1	2 Gb	32	256 MB
LPDDR2	x16	2	2 Gb	32	512 MB
LPDDR2	x16	1	2 Gb	16	256 MB

10.1.6 I/O Signals

The DDR signal pins are listed in Table 10-3. The DDR I/O buffers are powered by the VCC_DDR power pins.

Note: The 7z010 supports only the 16-bit data bus width, not the 32-bit bus.

Table 10-3: DDR I/O Signal Pin List

Device Pin Name	I/O	Connections			Description
		DDR2	LPDDR2	DDR3	
DDR_CK_{P,N}	O	X	X	X	Differential clock outputs
DDR_CKE	O	X	X	X	Clock enable
DDR_CS_B	O	X	X	X	Chip select
DDR_RAS_B	O	X		X	RAS row address strobe
DDR_CAS_B	O	X		X	RAS column address strobe
DDR_WE_B	O	X		X	Write enable
DDR_BA[2:0]	O	X		X	Bank address
DDR_A[14:0]	O	X	X	X	DDR2/DDR3: Row/Column Address LPDDR2: CA[9:0] = DDR_A[9:0]
DDR_ODT	O	X		X	Output dynamic termination signal
DDR_DRST_B	O			X	Reset
DDR_DQ[31:0]	IO	X	X	X	32-bit Data bus: [31:0] 16-bit Data bus: [15:0] 16-bit Data with ECC
DDR_DM[3:0]	O	X	X	X	Data byte masks
DDR_DQS_{P,N}[3:0]	IO	X	X	X	Differential data strobes

Table 10-3: DDR I/O Signal Pin List

Device Pin Name	I/O	Connections			Description
		DDR2	LPDDR2	DDR3	
DDR_VR{P,N}	~	X	X	X	DCI voltage reference. Used to calibrate input termination, and DDR I/O drive strength. Connect DDR_VRP to a resistor to GND. Connect DDR_VRN to a resistor to VCC_DDR.
DDR_VREF{0,1}	~	X		X	Voltage reference

10.2 AXI Memory Port Interface (DDRI)

10.2.1 Introduction

Each AXI master port has an associated slave port in the arbiter. The command FIFO located inside the port stores the address, length and ID contained in the command. The RAM in the write port stores the write data and byte enable. The RAM in the read port stores the read data coming back from the core.

Because the read data coming back from the core can come out of order, the RAM is used for data re-ordering. Each AXI command can make a request (write or read) for up to 16 data transfers (up to the AXI limit). A single command coming from the AXI can be split into multiple requests going to the arbiter logic and the controller.

The incoming command is first stored in the command FIFO. After a valid command is detected in the write or read port, the value of the length field is checked and the number of requests associated with this command is calculated. The logic then sends arbiter requests to the arbitration logic. The arbitration logic looks at the requests from all the ports and gives the grant to one port at a time.

When a write port receives the grant from the arbiter, it generates write address, and write data pointer and asserts the command valid. A read port on receiving grant generates read address, read command length and the read token and asserts the command valid. Requests from various ports are multiplexed using the grant signal.

When a write command is accepted by the DDR controller, it sends the write data pointer back to the arbiter. The write data from all ports is multiplexed using the port ID contained in the write data pointer.

When the read data comes back from the core, an associated ID is used direct the data to the appropriate read port. According to AXI specifications, the read data with the same ID is required to be given back to the AXI read master in the same order in which read commands were received by the port.

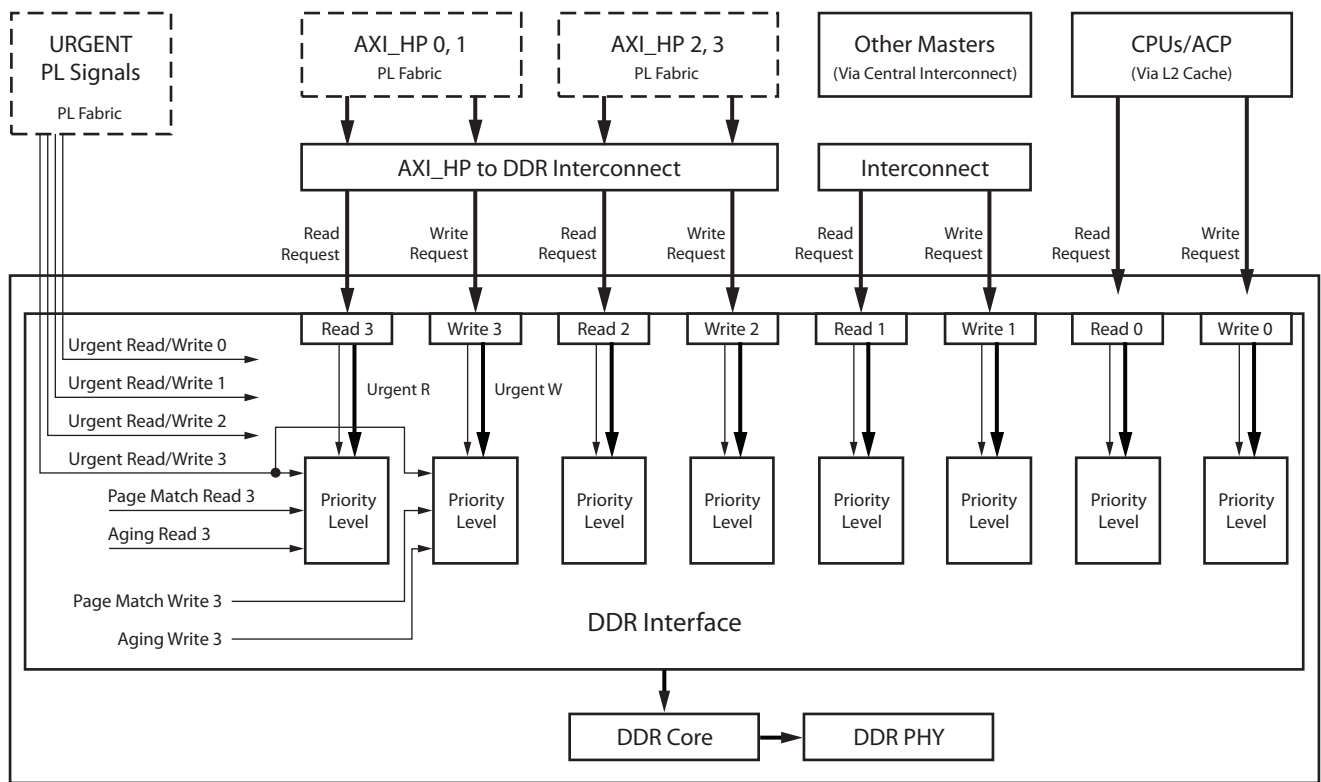
DDRI Features:

- Four identical 64-bit AXI ports support INCR and WRAP burst types
- Sophisticated arbitration schemes to prevent data starvation
- Low latency path using urgent bit to bypass arbitration logic

- Deep read and write command acceptance capability
- TrustZone extension with 64 MB resolution
- Out-of-order read data returned for requests with different master ID
- Nine-bit AXI ID signals on all ports
- Burst length support from 1 to 16 data beats
- Burst sizes of 1, 2, 4, 8 (bytes per beat)
- Does not support locked accesses from any AXI port

10.2.2 Block Diagram

The block diagram of the DDRI is shown in Figure 10-3.



UG585_c10_03_032012

Figure 10-3: DDRI Block Diagram

10.2.3 AXI Feature Support and Limitations

This list shows supported and unsupported features for the AXI ports into the DDRI:

- Fixed burst type is not supported. Note that the behavior is unknown if this transfer type is received at one of the AXI ports.
- Byte, half-word and word sub-width commands are supported.

- EXCL accesses are only supported on a single DDR port, ie., there is no support for EXCL accesses across DDR ports.
- AWPROT/ARPROT[1] bit is used for trust zone support, AWPROT/ARPROT[0], and AWPROT/ARPROT[2] bits are ignored and do not have any effect.
- ARCACHE[3:0]/AWCACHE[3:0] (cache support) are ignored, and do not have any effect.
- Sparse AXI write transfers (random strobes asserted/de-asserted for any data beat) are supported.
- Unaligned transfers are supported.

10.2.4 TrustZone

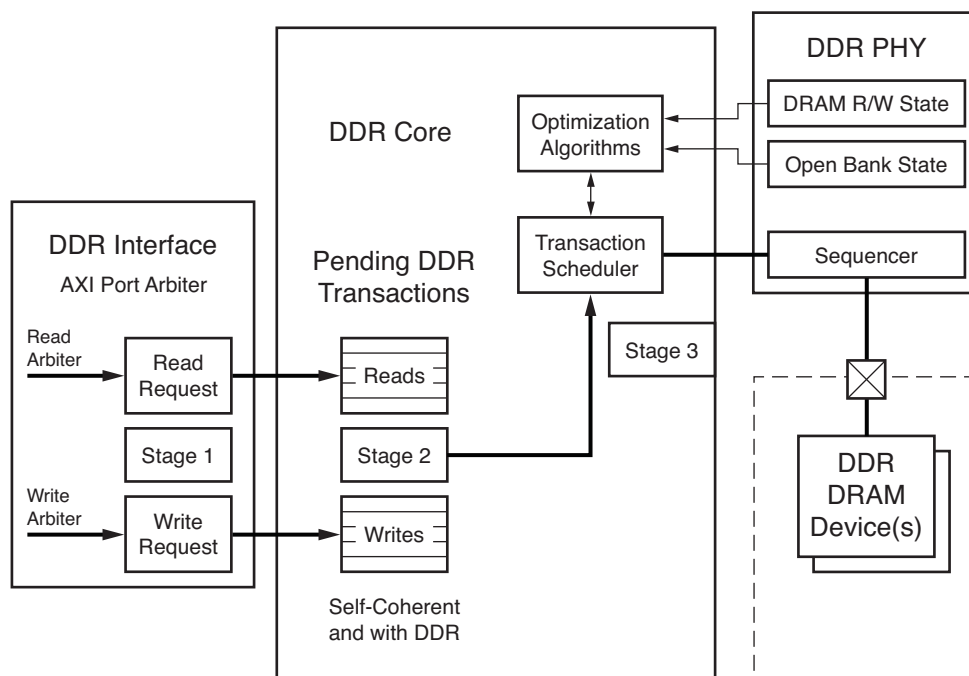
The DDR memory can be configured in 64 MB sections. Each section can be configured to be either secure or non-secure. This configuration is provided via a system level control register.

- A 0 on a particular bit indicates a secure memory region for that particular memory segment.
- A 1 on a particular bit indicates a non-secure memory region for that particular memory segment.

In the case of a non-secure access to a secure region, a DECERR response is returned back to the master. For writes, the write data is masked out before being sent to the controller which results in no actual writes occurring in the DRAM. On reads, the read data is all zeros on a TZ violation.

10.3 DDR Core and Transaction Scheduler (DDRC)

The DDRC is comprised of queues for pending read and write transactions and a scheduler that pops off the queues and sends the next transaction to the DDR PHY. Between the DDRI and the DDRC the arbitration of which transaction is sent to the DDR PHY and the DDR devices.



UG585_c10_04_032012

Figure 10-4: DDRC Block Diagram

10.3.1 Row/Bank/Column Address Mapping

The DDRC is responsible for mapping byte-addressable physical addresses used by the PS and PL AXI masters to DDR row, bank and column addresses. This address mapping has a limited configurability to allow user optimization. Optimizing the mapping to specific data access patterns can allow increased DDR utilization by reducing page and row change overhead.

Note: Many combinations of address remapping are not available, notably a complete bank-row-column mapping.

The address mapper associates linear request addresses to DRAM addresses by selecting the AXI bit that maps to each and every applicable DRAM address bit. The full available address space is only accessible to the user when no two DRAM address bits are determined by the same AXI address bit.

Each DRAM row, bank, and column address bit has an associated register vector to determine its associated AXI source in the DDRC `DRAM_addr_map_bank`, `DRAM_addr_map_row`, and `DRAM_addr_map_col` registers. The associated AXI address bit is determined by adding the internal base of a given register to the programmed value for that register, as described in the following equation:

$$[\text{internal base}] + [\text{register value}] = [\text{AXI address bit number}]$$

For example, from the description for `reg_ddrc_addrmap_col_b3`, it can be seen that this register determines the mapping for DRAM column bit 4 and its internal base is 3. When the full data bus is in use, DRAM column bit 4 is determined by the following:

$$[\text{internal base}] + [\text{register value}].$$

If `reg_ddrc_addrmap_col_b3` register is programmed to 2, then the AXI address bit is:

$$3 + 2 = 5.$$

In other words, the column address bit 4 sent to DRAM is mapped to AXI address bit `*_ADDR[5]`.

All the column bits left-shift one bit in half bus width mode (including ECC). In this case, `reg_ddrc_addrmap_col_b2` determines the mapping of DRAM column address bit 4. In the full bus width case, `reg_ddrc_addrmap_col_b3` determines DRAM column address 4.

10.4 DDRC Arbitration

The DDRC arbitration consists of three stages (see [Figure 10-5](#)):

- Stage 1 is AXI read/write port arbitration
- Stage 2 is winner of read and write compete
- Stage 3 is transaction scheduler

Each of these stages has their own arbitration steps that will be discussed in more detail.

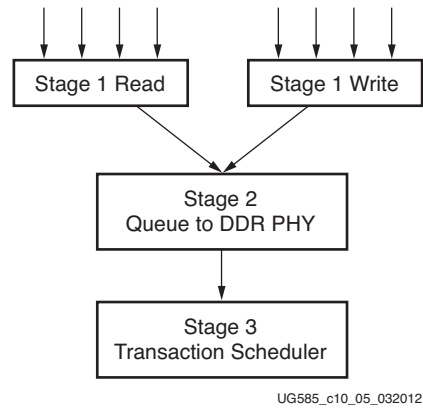


Figure 10-5: DDC Arbitration

10.4.1 Priority, Aging Counter and Urgent Signals

DDR controller arbitration is based on round robin with aging. The round robin mechanism circularly scans all requesting devices and services all outstanding requests before servicing the same device again. The aging mechanism measures the time each request has been pending and assigns higher priority to requests with longer wait times.

Each of the DDRC read and write ports is assigned a 10-bit priority value (see registers `axi_priority_wr_port0-3` and `axi_priority_rd_port0-3`). This value is used as an initial value for an aging counter that counts down. Thus at any instant, a lower aging counter value takes priority over a higher one.

In addition, each of the DDRC read and write ports has an *urgent* input signal. This signal acts as a reset to the aging counter. When urgent is asserted, the aging counter for that port is reset, instantly making this port's priority the highest. The source of the urgent bit is selectable via an SLCR host-programmable register (`DDR_URGENT_SEL`) to be one of the following:

- The most-significant bit of the 4-bit QoS signal in the AXI interface for a port (except for memory port 0 used by the CPUs and APU)
- A programmable SLCR register value (`DDR_URGENT`)
- One of the PL signal `DDRARB[3:0]` bits

While the priority value is static in nature, the urgent bit and QoS signal can be manipulated dynamically.

10.4.2 Page-Match

To improve DDR utilization, the address of each new request is compared with the address of the previous request. The DDRC has a preference for taking new requests that are to the same page as the previous request. The memory port compares the addresses to determine if there is a page match. A port that has been selected by the arbiter continues to get preference (priority 0) as long as there continues to be page hits. They will compete against other ports with a priority of 0.

The page size is defined by PAGE_MASK (32 bit register that all bits are the mask) and is always address aligned. For proper operation, the software must program the page size in the PAGE_MASK to match the size of the DDR memory. Setting this register to 0 disables the page-match step of the arbitration.

10.4.3 Aging Counter

When a request is pending and not serviced, a decrementing aging counter is enabled. The starting value of this counter is loaded from the 10-bit value in the priority register (axi_priority_<rd/wr>_port<n>, there are 8 registers, one for each ports). The counter reloads when the request is serviced. The value of this counter is used to help indicate the priority of an AXI memory port. The lower the value of this counter, the higher the priority. When the priority reaches 0, the request has the highest priority.

For arbitration purposes, only the upper-most 5 bits are used to differentiate priority among ports. This keeps the arbitration mechanism to a manageable size and latency, while still comprehending an approximation of the age-based priority of each port.

10.4.4 Stage 1 – AXI Port arbitration

The eight ports (four read and four write) compete to get the DDRC to accept their request. The arbiter grants a request based on a many factors. Read and write requests are treated the same, meaning they go through the same arbitration. Each port maintains a priority level that steadily moves from a preset state to the highest state or 0. This mechanism is important to maintain a minimum bandwidth on a port. Each port also has different ways to signal an urgent situation, either on a per-transaction basis (QoS) or for multiple transactions. The per-transaction urgency can be good for low-latency masters.

The priority as shown in [Figure 10-6](#) has the following logic. If there is a port with Priority 0 or 0 in its aging counter (the highest priority), then it wins. If there is no port with 0 priority, the arbitration checks if the port being serviced has a page match. If there is no page match, the lowest value in the aging counter wins. If there is a tie for the lowest value in the aging counter, round robin is used to resolve any ties.

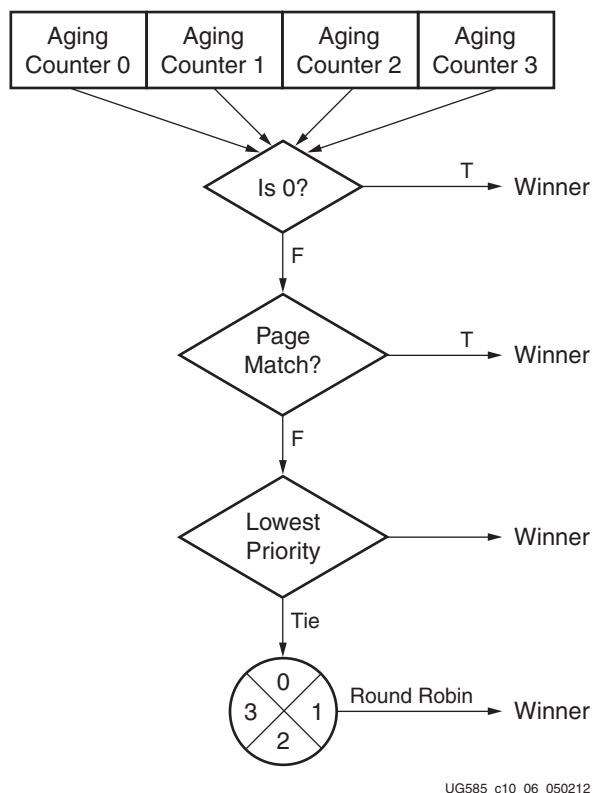


Figure 10-6: Stage 1 – AXI Port Arbitration

10.4.5 Stage 2 – Read Versus Write

The reads and the writes each have a queue in the DDR Core. The entries in these queues then vie for the next level of arbitration, shown in [Figure 10-7](#).

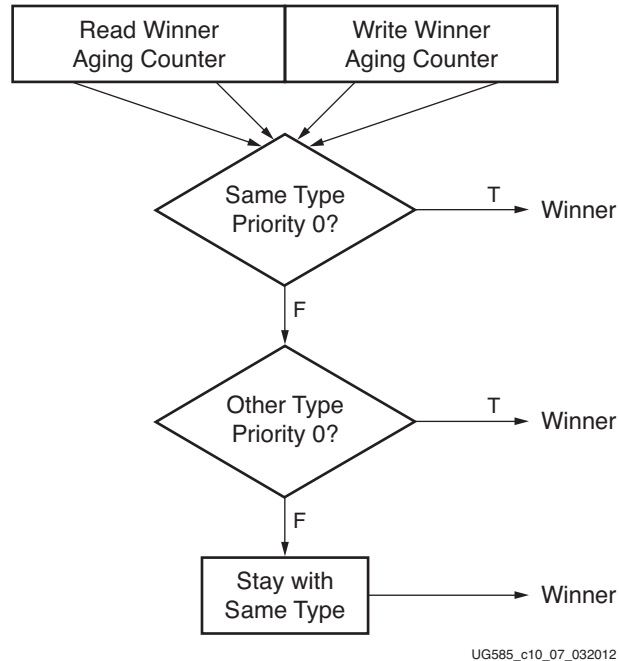
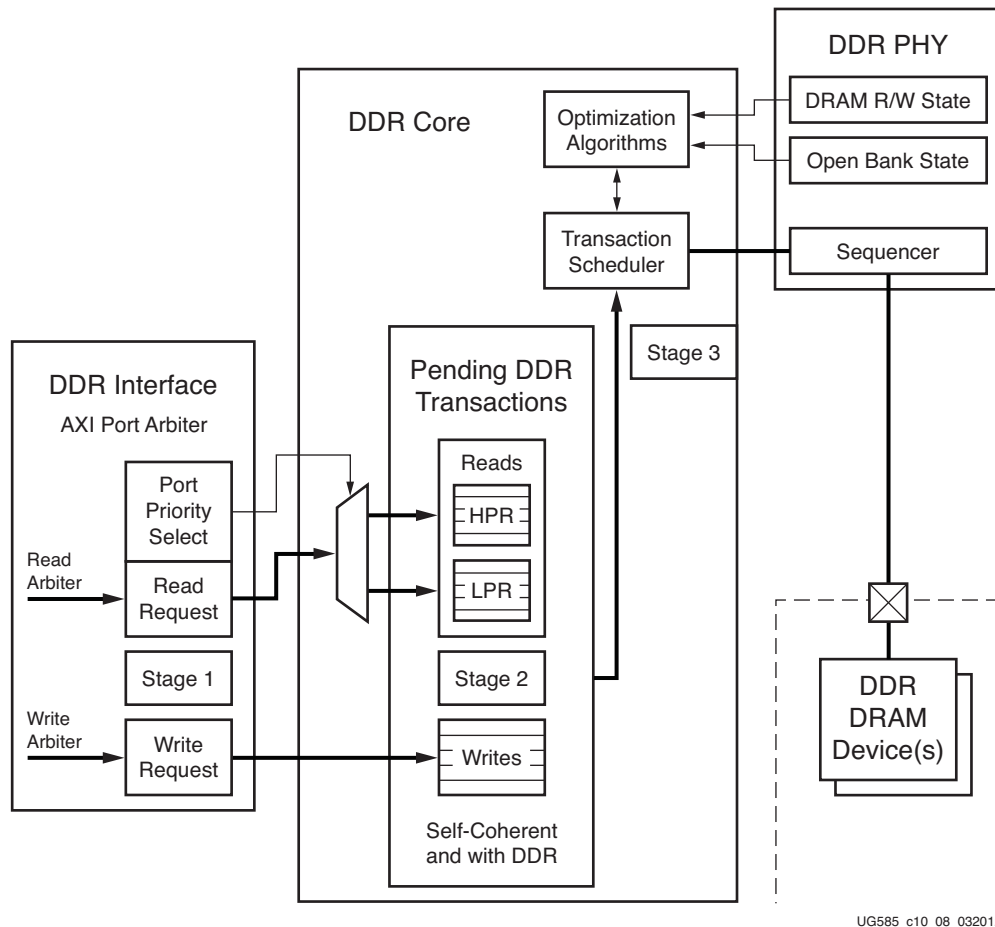


Figure 10-7: Stage 2 – Read Versus Write

This stage of arbitration starts with the aging counter as shown in [Figure 10-7](#). If there is a same type of transaction with a priority 0, it wins. For example if a read won the last round of arbitration and there is a read with priority 0, it wins. If there is not a same type of transaction with priority 0, and another type of transaction with a priority 0 is present, it wins. If there is no Priority 0 in the queue then it stays with the same type of transaction.

10.4.6 High Priority Read Ports

Before going into Stage 3 of the arbitration, a feature of the DDRC, the high priority read, needs to be described. The HPR, or high priority read feature, allows splitting the read data queue (32 words) within the DDRC into two separate queues for low and high priority. Each of the four read ports can be assigned a low or high priority. By default this feature is disabled. When used, a high priority read device is not slowed down by the (potentially slower) read data rate of a low-priority device. In a typical use case, HPR is enabled on port 0 (CPU), thus reducing the CPU average read latency. The split of the read data queue does not have to be two equal parts. Thus giving the CPU a small queue to bypass the larger queue to service reads that need a lower latencies. [Figure 10-8](#) shows where the read queue is split.



UG585_c10_08_032012

Figure 10-8: Read Queue

This can be changed by setting the `reg_arb_set_hpr_rd_port<n>` bit to 1'b1 for AXI ports (this is in the `axi_priority_<rd/wr>_port<n>` register). The DDRC is configured by default to serve only LPR. The read CAM can service only LPR by default. The total CAM depth is 32 for Read. (However, one slot is always allocated for ECC purposes.) The `reg_ddrc_lpr_num_entries` register field in the DDRC `ctrl_reg1` register specifies the number of entries reserved for LPR. Taking 31 and subtracting the `reg_ddrc_lpr_num_entries` gives the number of entries reserved for HPR.

It is necessary to change the `REG_DDRC_LPR_NUM_ENTRIES` field if a port is configured as an HPR port to avoid deadlock in the credit mechanism

10.4.7 Stage 3 – Transaction State

The transaction state is the last stage of arbitration before the transaction goes to the DDR PHY and the DDR device. The transaction state can be read or write. To change the transaction state there must be no more transactions of that type or there can be a critical transaction of the other type.

Figure 10-9 shows the simple state machine for this.

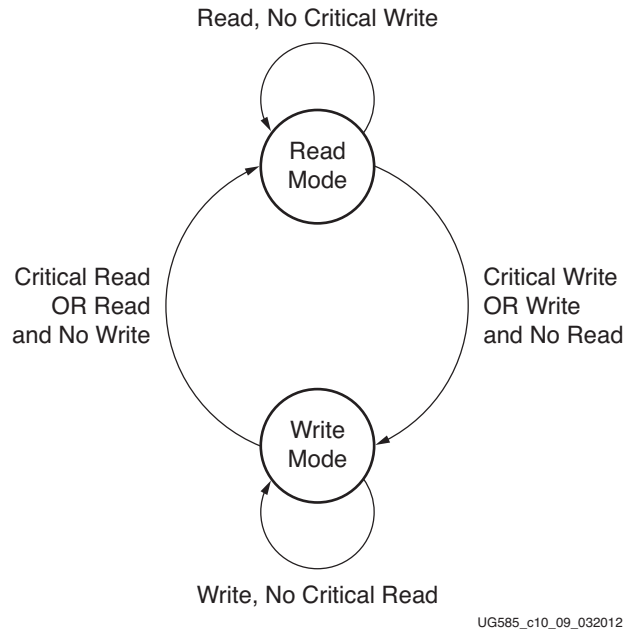


Figure 10-9: Stage 3 – Transaction State

The transaction state stays the same until the other type of transaction is critical or there is no more of that type of transaction. The state machine defaults to the read state. Table 10-4 shows how a transaction in the queue can go from a normal state to critical.

Table 10-4: Transaction Store State Transitions

Normal	Critical	A transaction has been pending for this transaction store and has not been serviced for a count of *_max_starve_x32 pulses of the 32-cycle timer.
Critical	Hard Non-Critical	*_xact_run_length number of transactions has been serviced from this transaction store.
Hard Non-Critical	Normal	*_min_non_critical number of cycles has passed in this state.

Notes:

* Can be WR, LPR or HPR. Example is wr_max_starve_x32 which is a field of the WR_Reg.

Taking the low priority read transaction store as an example, it is expected that the transaction store generally functions independently based on the following signals:

- lpr_max_starve_x32
- lpr_xact_run_length
- lpr_min_non_critical

The reg_arb_go2critical_en field in the DDRC ctrl_reg2 register enables the arbiter to drive co_gs_go2critical_* signals to the DDRC. There are sideband signals on AXI (awurgent and arurgent) that drive the co_gs_go2critical_* signals. If any port asserts their urgent sideband signal, and if this feature is enabled, the arbiter asserts the corresponding co_gs_go2critical_* signal to the controller.

Inside the controller, assertion of this signal causes the state machine to switch from one state to another. For example, if the DDRC is currently servicing reads and `co_gs_go2critical_wr` goes High, the controller ignores the normal state switching methods (starvation counter etc), and jumps to servicing writes. There is a register in the controller to control how long to keep servicing the current command type before switching to the other (`reg_ddrc_go2critical_hysteresis` field in the DDRC `ctrl_reg2`).

In summary, this `go2critical` feature is used in the controller and ensures fast switching between reads and writes for transactions with super high priority.

Note:

1. The normal programming condition is expected to be `reg_ddrc_prefer_Write=0`. (this is a bit field in the `DRAM_param_reg4` register) This means that the read requests are always serviced immediately when received by an idle controller. Also, it is often desirable to set the `reg_ddrc_rdwr_idle_gap` (this field is in the `ddrc_ctrl` register) to a very low number (such as 0, 1, or 2) to ensure that writes do not go un-serviced in an otherwise-idle controller for any length of time, wasting bandwidth. (The trade-off here is that by servicing writes more quickly, the likelihood increases that reads issued to the controller immediately following writes incurs additional latency to allow writes to be serviced and turn the bus around.)
2. Because the ordering is guaranteed on all requests issued to the controller, write latency must not be a concern to system design. (In the event that write data is required by a subsequent read, the controller automatically forces the write data out to DRAM before servicing the read.)

10.4.8 Read Priority Management

Normally in a read mode, high priority read requests are preferred for service over low priority read requests. However, if the low priority read transaction store is critical and the high priority read transaction store is not, then low priority read requests are preferred over high priority read requests. This prevents starvation of low priority reads.

10.4.9 Write Combine

The write combine feature allow multiple writes to the same address to be combined into a single write to DRAM. When a new write collides with a queued write in the CAM:

- If write combine is enabled, the DDRC overwrites the data for the old write with that from the new write and only performs one write transaction (write combine).
- If write combine is disabled, the DDRC follows the following sequence of operations:
 - Holds the new write transaction in a temporary buffer
 - Applies flow control back to the core to prevent more transactions from arriving
 - Flushes the internal queue holding the colliding transaction until that transaction has been serviced
 - Accepts the new transaction and removes flow control

10.4.10 Credit Mechanism

The DRAM controller employs a credit mechanism to ensure that buffers do not overflow (pending DDR transactions). The interface making the request to the controller can only request commands for which it has been granted credits or open slots in the queues to issue.

Credits are tracked separately for the following three command types:

- High priority reads
- Low priority reads
- Writes

Credits are counted for each command type independently according to the following rules:

- Initially the interface has zero credits.
- Following the de-assertion of reset to the DRAM controller, credits are issued to the interface for each command type. A given credit count increments every time a credit is issued by the DRAM controller, indicated by the assertion of the appropriate *_credit signal on the rising edge of the clock.
- When the credit count is greater than zero, the interface can issue requests of that type to the controller. Each time a request is issued to the controller, the associated credit count is decremented.

10.5 Controller PHY (DDRP)

The DDRP processes read and write requests from the DDRC and translates them into specific signals within the timing constraints of the target DDR memory. The DDRP is composed of functional units including PHY control, master DLL, and read/write leveling logic. The PHY data slice block handles the DQ, DM, DQS, DQ_OE and DQS_OE signals. The PHY control block synchronizes all of the control signals with the DDR_x3 clock.

There are two kinds of DLLs, the master DLL, and the slave DLL. The DLLs are responsible for creating the precise timing windows required by the DDR memories to read and write data. The master DLL measures the cycle period in terms of a number of taps and passes this number through the ratio logic to the slave DLLs. The slave DLLs can be separated on the target die to minimize skew and delay and to account for process, temperature and voltage variations.

Write leveling and read leveling are new functions required for DDR3 operation. These functions help automatically determine delay timings required to align data to the optimal window for reliable data capture:

- Read leveling and write leveling for DDR3
- Support for 16- and 32-bit data bus width with one rank
- Optional ECC with a 16-bit data width
- Individual bytes with read data mask bits

10.5.1 Loopback

The PHY has capabilities to perform loopback operation. This functionality is reserved.

10.6 Initialization and Calibration

To start operation of the PS DRAM interface, the following sequence of operations must take place:

1. DDR clock initialization
2. DDR I/O buffers (DDR IOB) initialization and calibration
3. DDR controller (DDRC) register programming
4. DRAM reset and initialization
5. DRAM input impedance (ODT) calibration
6. DRAM output impedance (Ron) calibration
7. DRAM Training
 - a. Write leveling
 - b. Read DQS gate training
 - c. Read data eye training

Note: This section is intended for reference and debug purposes only. Generally, programming for steps 1–7 are provided via the appropriate Xilinx design suite, such as EDK XPS.

10.6.1 DDR Clock Initialization

Prior to DDR initialization, a DDR clock must be active. Both the `ddr_2xclk` and `ddr_3xclk` clocks must be configured properly. The `ddr_3x clk` is the clock used by the DRAM and should be set to the desired operating frequency (note that the data rate per bit is twice the operating frequency). The `ddr_2xclk` is used by the interconnect and is typically set to 2/3 of the operating frequency. The DDR PLL frequency should be set to an even multiple of the operating frequency.

Table 10-5 provides frequency configuration examples assuming a 50 MHz reference clock.

Table 10-5: Frequency Configuration Examples

Operating Frequency	DDR PLL Frequency	ddr_3xclk Frequency	ddr_2xclk Frequency	PLL Feedback Divider	ddr_3xclk Divider	ddr_2xclk Divider
525.00	1050.00	525.00	350.00	21	2	3
400.00	1600.00	400.00	266.67	32	4	6

Programming the DDR clock involves the DDR_PLL_CTRL and DDR_CLK_CTRL registers in the SLCR. The programming sequence is as follows.

1. Put the PLL in bypass mode
2. Assert the PLL reset
3. Set the PLL feedback divider value
4. Set the ddr_2xclk and ddr_3xclk divisor values
5. Deassert the PLL reset
6. Wait for PLL lock
7. Remove PLL bypass mode

In addition to the main DDR clock, a 10 MHz clock is used by the digitally controlled impedance (DCI) function built into the DDR IOB. This clock is configured via the SLCR DCI_CLK_CTRL register.

10.6.2 DDR IOB Impedance Calibration

The DDR IOBs support calibrated drive strength and termination strength using the DCI digitally controlled impedance mode of the IOB. In DDR2/DDR3 modes this is used to calibrate termination strength. In LPDDR2 mode this is used to calibrate drive strength. The DCI state machine requires two external pins, VRN and VRP, which are connected to external resistors to V_{CCO_DDR} and ground, respectively. DCI settings are shown in [Table 10-6](#).

Table 10-6: DCI Settings

DDR standard	Termination Impedance	Drive Impedance	VRN resistor (to V _{CCO_DDR})	VRP resistor (to ground)
DDR3	40Ω		80Ω	80Ω
DDR2	50Ω		100Ω	100Ω
LPDDR2	None	40Ω	40Ω	40Ω

When enabled, the DCI state machine will automatically match drive and termination impedance to the external resistors. This background calibration takes 1-2 ms to lock and then runs continuously.

Calibration

1. Configure the clock module to configure a 10 MHz clock on dci_clk
2. Enable the DDR DCI calibration system using the SLCR registers DDRIOB_DCI_CTRL and DDRIOB_DCI_STATUS
 - a. Toggle DDRIOB_DCI_CTRL.RESET_B to 0 and set to 1
 - b. Set DDRIOB_DCI_CTRL.PREF_OPT, and NREF_OPT fields according to [Table 10-7](#)
 - c. Set DDRIOB_DCI_CTRL.UPDATE_CONTROL to 0
 - d. Set DDRIOB_DCI_CTRL.ENABLE to 1
 - e. Poll on the DDRIOB_DCI_STATUS.DONE bit until it is 1

Table 10-7: Calibration

Field Name	Reset	Down Power	DCI Enable DDR3	DCI Enable DDR2	DCI Enable LPDDR2	DCI Disabled
UPDATE_CONTROL	0	0	1	1	1	1
PREF_OPT2[2:0]	000	000	000	000	000	000
PREF_OPT1[1:0]	00	00	00	00	10	00
NREF_OPT4[2:0]	000	000	001	001	001	000
NREF_OPT2[2:0]	000	000	000	000	000	000
NREF_OPT1[1:0]	00	00	00	00	10	00

10.6.3 DDR IOB Configuration

The DDR IOBs must be configured to function as I/O. Each type of DDR IOB is controlled by two different SLCR configuration registers. The configuration registers configure the IOB's input mode, output mode, DCI mode, and other functions.

Configuration

The DDR system supports DDR3/DDR2/LPDDR2 in 16 and 32 bit modes and power down modes. The registers identified in Table 10-8 control groups of I/Os and must be configured depending on the particular mode.

Table 10-8: DDR IOB Configuration Registers

Register	Affected I/O Blocks	Description
DDRIOB_DDR_CTRL	VREF, VRN, VRP, DRST	Controls special I/O modes for internal and external VREF and DCI reference pins VRN and VRP
DDRIOB_DCI_CTRL	DCI controller	Enables the DCI controller
DDRIOB_DCI_STATUS	DCI controller	Status for the DCI controller
DDRIOB_ADDR0 DDRIOB_ADDR1	DDR_A[14:0], DDR_CKE, DDR_BA[2:0], DDR_ODT, DDR_WE_B, DDR_CAS_B, DDR_RAS_B, DDR_CS_B	Configuration settings for address and control outputs used by LPDDR2, DDR2 and DDR3
DDRIOB_CLOCK	DDR_CK_P, DDR_CK_P	Configuration settings for the differential clock outputs. Controls DDR_CK_P, DDR_CK_P
DDRIOB_DATA0	DDR_DQ[15:0], DDR_DM[1:0], DDR_FIFO_IN[0], DDR_FIFO_OUT[0]	Configuration settings for data and mask bits for lower 16-bits
DDRIOB_DATA1	DDR_DQ[31:16], DDR_DM[3:2], DDR_FIFO_IN[1], DDR_FIFO_OUT[1]	Configuration settings for data and mask bits for upper 16-bits
DDRIOB_DIFF0	DDR_DQS_P[1:0], DDR_DQS_N[1:0]	Configuration settings for dqs bits for lower 16-bits
DDRIOB_DIFF1	DDR_DQS_P[3:2], DDR_DQS_N[3:2]	Configuration settings for dqs bits for upper 16-bits
DDRIOB_DRIVE_SLEW_ADDR	DDR_A[14:0], DDR_CKE, DDR_BA[2:0], DDR_ODT, DDR_WE_B, DDR_CAS_B, DDR_RAS_B, DDR_CS_B	Drive strength and slew rate settings for address and control output
DDRIOB_DRIVE_SLEW_CLOCK	DDR_CK_P, DDR_CK_P	Drive strength and slew rate settings for the clock outputs

Table 10-8: DDR IOB Configuration Registers (Cont'd)

Register	Affected I/O Blocks	Description
DDRIOB_DRIVE_SLEW_DATA	DDR_DQ[31:0], DDR_DM[3:0], DDR_FIFO_IN[1:0], DDR_FIFO_OUT[1:0]	Drive strength and slew rate settings for data I/Os
DDRIOB_DRIVE_SLEW_DIFF	DDR_DQS_P[3:0], DDR_DQS_N[3:2]	Drive strength and slew rate settings for data strobe I/Os

Set the IOB configuration as follows:

1. Set DCI_TPYE to DCI_DRIVE for all LPDDR2 I/Os
2. Set DCI_TPYE to DCI_TERM for DDR2/DDR3 bidirectional I/Os
3. Set OUTPUT_EN = obuf to enable outputs
4. Set TERM_DISABLE_MODE and IBUF_DISABLE_MODE to enable power saving input modes
5. Set INP_TYPE to V_{REF} based differential receiver for SSTL, HSTL for single ended inputs
6. Set INP_TYPE to Differential input receiver for differential inputs
7. Set TERM_EN to enable for DDR3 and DDR2 bidirectional I/Os (Outputs and LPRDDR2 IOs are un terminated)
8. Set data1, diff1 to power down if 16 bit DDR is used
9. For DDR2 and DDR3 – DCI only affects termination strength, so address and clock outputs do not use DCI.
10. For LPDDR2 – DCI affects drive strength, so all I/Os use DCI.

VREF Configuration

DDR I/Os use a differential input receiver. One input to this receiver is connected to the data input, and the other is connected to a voltage reference called V_{REF} . For DDR2/3 and LPDDR2 DRAM interfaces, the V_{REF} voltage is set to half of the I/O V_{CCO} voltage. The V_{REF} can be supplied either externally over dedicated V_{REF} pads, or from an internal voltage source. To configure the V_{REF} reference supply, set the DDRIOB_DDR_CTRL register as follows:

- To enable internal V_{REF}
 - Set DDRIOB_DDR_CTRL.VREF_EXT_EN to 00 (disconnect I/Os from external signal)
 - Set DDRIOB_DDR_CTRL.VREF_SEL to the appropriate voltage setting depending on the DDR standard ($V_{vref}=V_{vcco_ddr}/2$)
 - Set DDRIOB_DDR_CTRL.VREF_INT_EN to 1 to enable the internal V_{REF} generator
- To enable external V_{REF}
 - Set DDRIOB_DDR_CTRL.VREF_INT_EN to 0 to disable the internal V_{REF} generator
 - Set DDRIOB_DDR_CTRL.VREF_SEL to 0000
 - Set DDRIOB_DDR_CTRL.VREF_EXT_EN to 11 to connect the IOBs V_{REF} input to the external pad. for a 32-bit interface
 - Set DDRIOB_DDR_CTRL.VREF_EXT_EN to 01 to connect the IOBs V_{REF} input to the external pad for a 16-bit interface

10.6.4 DDR Controller Register Programming

Prior to enabling the DDRC, all DDRC registers must be initialized to system-specific values. About 80 registers with over 350 parameters might be set or left at their power-on default values. The DDRC is then enabled, by writing to the `ddrc_ctrl` register. Once enabled, the DDRC automatically performs the initialization steps 4-7 ([Initialization and Calibration](#)). DDRC operation is autonomous, requiring no further programming unless functionality changes are desired (e.g. changing AXI port priority levels).

10.6.5 DRAM Reset and Initialization

The DDRC performs DRAM reset and initialization per the JEDEC specs, including reset, refresh, and mode registers initialization.

10.6.6 DRAM Input Impedance (ODT) Calibration

The on-die-termination (ODT) is available in DDR2 and DDR3 devices with the following features:

- In DDR3 devices, the ODT value is controlled via Mode register MR1. It can be disabled, or set to one of the following values: 120Ω, 60Ω, or 40Ω.
- In DDR2 devices, the ODT value is controlled via the mode register EMR. It can be disabled, or set to one of the following values: 75Ω, 150Ω, or 50Ω.
- Both DDR2 and DDR3 devices have a dedicated ODT input pin that is used to enable the ODT during write operations, and disable it otherwise.

Calibration

DDR3 devices provide ODT calibration via the ZQCL and ZQCS commands. The ZQCL (ZQ calibration long) command is issued as part of the DRAM initialization procedure and is used for initial calibration, which takes about 512 DDR_3x clock cycles. The ZQCS (ZQ calibration short) is subsequently issued automatically by the DDRC for minor calibration adjustments. A typical ZQCS interval is 100 ms.

DDR2 (and LPDDR2) devices do not provide ODT calibration.

10.6.7 DRAM Output Impedance (R_{ON}) Calibration

The output impedance control feature is available in DDR2, DDR3 and LPDDR2 devices.

- In DDR2 devices, the value is controlled via the mode register EMR, and can be set to full strength or reduced strength.
- In DDR3 devices, the value is controlled via the mode register MR1, and can be set to one of the following values: 40Ω or 35Ω.
- In LPDDR2 devices, the value is controlled via MR3, and can be set between 34Ω and 120Ω (default value is 40Ω).

Calibration

In DDR3 and LPDDR2 devices, the output impedance is calibrated by the same ZQCL/ZQCS commands discussed above.

In DDR2 devices, the DDR2 external calibration procedure (OCD for off-chip driver calibration) is not supported by the DDRC.

10.6.8 DRAM Training

DRAM training includes three steps, executed in the following order:

1. Write leveling
2. Read DQS gate training
3. Read data eye training

Not all DRAM types support all three steps, as detailed below. Each step can be enabled or disabled independently.

If a training step is enabled, the user must provide an initial delay value as a starting point of the automatic training procedure. The value is a rough estimate of the expected delay or skew (see details below) on the system board, minus some margin.

If a training step is disabled, the user must provide a delay value to be used to compensate for the board delay or skew.

There are several possible reasons why the user might choose to disable a training step.

- The step is not supported by the particular DRAM type. For example, write leveling is not supported by DDR2 and LPDDR2.
- Board delays are well-known and operating conditions are such that timing variance is minimal, and training is not required.
- Delay settings are known from previous training events.

Training time is on the order of 1-2 ms at a 500 MHz DRAM clock.

Write Leveling

Goal	Adjust WR DQS relative to CLK
Desired Nominal	DQS aligned with clock (0 phase offset)
Final Ratio	Equal to the DQS to CLK board delay at the DRAM
Initial Ratio	Final value minus 0.5 cycle. If < 0 set to 0. If skew is too small, invert clock.
Applies To	DDR3 only

Write leveling is part of the DDR3 specification. Due to the fly-by topology recommended for DDR3 systems, the clock (CLK) tends to lag relative to write DQS at the DRAM input. In order to align CLK

and DQS as required by the DRAM specification, the PHY delays the DQS signal to match the board skew. The write leveling procedure is used to find the required delay.

When write leveling is enabled (via MR1), the DRAM asynchronously feeds back CLK, sampled with the rising edge of DQS, through the DQ bus. The controller repeatedly delays DQS until a transition from 0 to 1 is detected. Write leveling is performed independently for each byte lane.

The DDRC supports write leveling as part of the initialization procedure. Optionally, write leveling can be disabled and pre-determined delay values can be programmed via registers (required for DDR2 and LPDDR2 where write leveling is not supported).

It should be noted that successful training depends on providing an approximate minimum DQS to CLK delay value. This value should be estimated based on system board layout.

Read DQS Gate Training

Goal	Adjust valid RD DQS window.
Desired Nominal	Surround the 4 (BL=8) valid DQS pulses
Final Ratio	2 * board delay. Add 0.5 cycle if the clock is inverted
Initial Ratio	Final ratio minus 0.125 cycle (0x20 units), but not < 0.
Applies To	DDR3, LPDDR2

The read DQS gate training is used by the PHY to identify the valid interval of read DQS and capture the read data. It is necessary to align the valid read window to the read data burst and exclude the preamble period and any period during which the DQS signal is tri-stated or driven by the PHY itself.

The DDRC supports read DQS gate training as part of the initialization procedure. Optionally, training can be disabled and pre-determined delay values can be programmed via registers (required for DDR2, where read training is not supported).

It should be noted that successful training depends on providing an approximate minimum Zynq to DRAM board delay value. This value should be estimated based on system board layout.

Read Data Eye Training

Goal	Adjust RD DQS relative to RD data.
Desired Nominal	DQS edge in the middle of data eye
Final Ratio	Nominal ideal value is 0.25 cycle since at DRAM output DQ and DQS are aligned
Initial Ratio	None required
Applies To	DDR3, LPDDR2

Read Data Eye Training Summary Table

Enabled by the MPR bit-field in MR3, DDR3 Read data eye training is done to compensate for possible imbalanced loading on the read path. In this mode, the DRAM outputs a stream of 01010101 in a burst length of 8 bits with a regular memory read command. Given the known data

pattern, the memory controller adjusts the internal DQS delay so that DQS edges occur in the middle of the data eye.

The DDRC supports read data eye training as part of the initialization procedure. Optionally, training can be disabled and pre-determined delay values can be programmed via registers (required for DDR2 where read training is not supported).

10.6.9 Write Data Eye Adjustment

There is no DDRC support for write data eye training, i.e., automatic alignment of write data relative to write DQS (recall that write leveling adjusts write DQS relative to CLK). However, manual alignment is possible.

Nominally, write DQS edges should be aligned in the middle of the write data eye at the DRAM inputs. The DDRC PHY provides a user-programmable phase shift value of data relative to DQS. The default nominal value is a 90 degrees phase shift. Given a balanced board design in which the DQ and DQS signals exhibit the same delay and loading, the default value is adequate. Otherwise, the user can provide a different phase shift value.

10.6.10 ECC Initialization

ECC is supported in 16-bit bus mode only. When enabled, a write operation computes and stores an ECC code along with the data, and a read operation reads and checks the data against the stored ECC code. It is therefore possible to receive ECC errors when reading uninitialized memory locations. To avoid this problem, all memory locations must be written before being read. Note that, since ECC is computed and checked over a byte resolution, a read of 1 byte is done to a 16-bit location that has only that byte initialized (second byte of 16-bit location is uninitialized) does not result in an ECC error. The controller only checks ECC on the byte that has been read.

Note that while only two data byte lanes are used for actual data, all four lanes are used in ECC mode, and therefore DDR training must be performed on all lanes.

10.6.11 Alternatives to Automatic DRAM Training

If for some reason the automatic training is not successful, alternative calibration schemes can also be used.

Training failures can be detected by performing a simple memory write-read-compare test. Since training is done independently for each byte lane, the memory test should check each data byte independently. In the event of training failure, two possible solutions are proposed here: a semi-automatic and a manual training method. As the method gets more manual, the training time increases. It is therefore recommended to follow this sequence:

1. Try automatic training, verify board measurement-driven initial values
2. If failed, try semi-automatic training
3. If failed, use manual training

Automatic Training

The standard training procedure is described above.

The estimated time for initialization and training is 1-2 ms.

Semi-Automatic Training

This method is useful when system/board delays are known, but PVT timing uncertainty causes the automatic training to fail. Note that only two initial timing parameters are needed to enable successful automatic training:

- Write DQS to CLK skew
- The one-way board delay from Zynq to DRAM

These values are known in this case, but the PHY PVT variations modify these values in an additive fashion. Therefore, given a nominal delay value T , the actual value might be in the range $(T-\delta, T+\delta)$, where δ is the maximum PVT variation.

The semi-automatic training method is performed as follows:

1. Divide the range $(T-\delta, T+\delta)$ into n parts, and thus create $(n+1)$ possible values for each of the two delay parameters.
2. Perform $(n+1)^2$ automatic training procedures and follow each one with a memory test.

For example, for $n=2$, the three data points for each parameter are $T-\delta$, T , and $T+\delta$. Perform nine automatic training procedures and observe the results. For $n=4$, perform 25 tests, etc.

As final parameters, pick the values that are in the center of the successful tests region. Note that each data byte lane (aka data slice) has its own independent parameters, and should be tested independently in the memory tests.

The estimated time for a training iteration is 1-2 ms plus the duration of the memory test. Assuming a simple 1,000 word read-write test and an average access time of 30 cycles, test duration is on the order of 60,000 cycles or about 0.12 ms at 500 MHz. Thus, a 25-iteration semi-automatic training might last 25-50 ms.

Multi-Set Semi-Automatic Training

Before resorting to manual training, a multi-set semi automatic training method is recommended.

The DDR PHY contains five adjustable delay elements, four of which are per byte lane (so the actual number of unique adjustable delay elements is 17). Of these five elements, only three are adjusted by the automatic training. These three elements are the write DQS delay, read DQS delay, and read data delay. The remaining two elements are the write data delay, and the control path delay, which take their value from a programmable register, and the value is not adjusted by the automatic training.

The automatic training process varies the delay of those three elements over a wide range, and the semi-automatic procedure increases that range. If both automatic and semi-automatic procedures fail, it is highly likely that one or both of the remaining two delay elements require adjustment.

Therefore, multiple sets of semi-automatic training procedures can be run, each set using different values of the two remaining delay values. Thus we still take advantage of the efficiency of the automatic training, and reduce the total number of experiments compared to all-manual training.

Manual Training

This method is useful when nothing is known, or if the semi-automatic method has failed. In its simplest form, this method consists of:

- Disabling the automatic training
- Performing a manual sweep of all delay parameters over their entire range. For each setting:
 - Initialize the DDRC with training disabled
 - Perform a memory test
- Keeping a scoreboard of results
- Locating the mid-point of all delay parameters (which might be different for each data lane)

The recommended delay increment value per iteration is 1/32 of a clock cycle, thus requiring 32 iterations to cover a one-cycle delay range per parameter.

The estimated time for a manual training iteration is 700 us (500 us are required as part of the DRAM reset/initialization procedure for DDR3) plus the duration of the memory test, or about 0.8 ms. Simplifying assumptions can be used to reduce the search range, but even then the number of iterations might be on the order of 1,000, bringing the manual training time to about one second.

[Table 10-9](#) provides summary of register values involved in manual training. All values are in units of 1/256 of a clock cycles (256 units = 1 clock cycle, 8 units = 1/32 of a clock cycle).

Table 10-9: Manual Training Register Summary

	Parameter	Register	Nominal Value	Minimum Suggested Search Range
1	Write DQS delay	wr_dqs_slave_ratio[9:0]	DQS to DCLK delay	0 -256
2	Write data delay	wr_data_slave_ratio[9:0]	DQS to DCLK delay + 64	64-320
3	Read DQS gate delay	fifo_we_slave_ratio[10:0]	2 * board delay	0-512
4	Read Data to DQS delay	rd_dqs_slave_ratio[9:0]	53, placing the DQS edges in the middle of the data eye	0 - 104 ⁽¹⁾
5	Control	ctrl_slave_ratio[9:0]	128 (64 for LPDDR2)	64-192 (32-96 for LPDDR2)

Notes:

1. Parameter 4 is an offset value relative to parameter 3.

10.7 Register Overviews

In general, the DDRC registers are static and can only be changed while the DDRC is in reset. However, there is a set of registers labeled as dynamic in their description that can be modified at anytime.

10.7.1 DDRI

Table 10-10 shows an overview of DDRI registers. There are no dynamic bit fields in the DDRI registers.

Table 10-10: DDRI Registers Overview

Function	Hardware Register Name
Arbitration	
	PAGE_MASK
	AXI_PRIORITY_{WR,RD}_PORT{0:3}
Misc	
	AXI_ID

10.7.2 DDRC

Table 10-11 shows an overview of DDRC registers.

Table 10-11: DDRI Registers Overview

Function	Hardware Register Name	Dynamic Bit Fields
Status		
	MODE_STS_REG	~
Transaction Scheduler	HPR_REG	
	HPR_REG	~
	LPR_REG	~
	WR_REG	~
DDR Protocol		
	DRAM_PARAM_REG0	[13:6]: t_rfc_min
	DRAM_PARAM_REG1	~
	DRAM_PARAM_REG2	~
	DRAM_PARAM_REG3	[20:16]: refresh_to_x32
	DRAM_PARAM_REG4	~
	DRAM_ODT_REG	~

Table 10-11: DDRI Registers Overview (Cont'd)

Function	Hardware Register Name	Dynamic Bit Fields
	ODT_DELAY_HOLD	~
	CTRL_REG1	[12]: selfref_en [8]: refresh_update_level
	CTRL_REG2	~
	CTRL_REG3	~
	CTRL_REG4	~
	MODE_REG_READ	~
	LPDDR_CTRL{0:3}	
	DFI_TIMING	
DDR Refresh		
	CHE_REFRESH_TIMER01	~
	CHE_T_ZQ	[16]: dis_auto_refresh
	CHE_T_ZQ_SHORT_INTERVAL	~
DDR Init		
	DRAM_INIT_PARAM	
	DRAM_EMR_REG	
	DRAM_EMR_MR_REG	
	DRAM_BURST8_RDWR	
	DRAM_DISABLE_DQ	[1]: dis_dq
Address Mapping		
	DRAM_ADDR_MAP_{BANK,COL,ROW}	
Power Reduction		
	DEEP_PWRDWN_REG	[0]: deeppowerdown_en
ECC		
	CHE_ECC_CONTROL	~
	CHE_{CORR,UNCORR}_ECC:	
	_LOG	~
	_ADDR	~
	_DATA_31_0	~
	_DATA_63_32	~
	_ECC_DATA_71_64	~
	CHE_ECC_STATS	~
	ECC_SCRUB	~
	CHE_ECC_CORR_BIT_MASK:	~

Table 10-11: DDRI Registers Overview (Cont'd)

Function	Hardware Register Name	Dynamic Bit Fields
	_31_0	
	_63_32	~

10.7.3 DDRP

Table 10-12 shows an overview of DDRP registers.

Table 10-12: DDRP Registers Overview

Function	Hardware Register Name	Dynamic Bit Fields
DDR Control		
	DDRC_CTRL	[]: soft_rstb []: powerdown_en
	TWO_RANK_CFG	[]: t_rfc_nom_x32
	PHY_CONFIG{0:3}	~
	PHY_CMD_TIMEOUT_RDDATA_CPT	~
Training		
	PHY_{WR,RD,GATE}_LVL_FSM	~
	PHY_INIT_RATIO{0:3}	~
	REG_{64,65}	~
	REG_{2C,2D}	~
	REG69_6C{0:3}	~
	REG6E_71{0:3}	~
DLL		
	DLL_CALIB	~
	PHY_CTRL_STS	~
	PHY_CTRL_STS_REG2	~
	PHY_DLL_STS{0:3}	~
	DLL_LOCK_STS	~
	WR_DATA_SLV{0:3}	~
	PHY_{RD,WR}_DQS_CFG{0:3}	~
	PHY_WE_CFG{0:3}	~
Others		
	PHY_RCVR_ENABLE	~
	PHY_DBG_REG	~

10.8 Error Correction Code (ECC)

There is optional ECC support in half-bus width (16-bit) data width configuration only.

Externally 26 bits of a DRAM DDR device are required, 16-bits for data and 10 bits for ECC. Each data byte uses an independent 5-bit ECC field. This mode provides single error correction and dual error detection. The ECC bits are interlaced with the data bits and unused bits as shown in Table 10-13.

Table 10-13: ECC Data Bit Assignments

DRAM DQ pin	Number of Pins	Function
DQ[7:0]	8	First Data Byte
DQ[15:8]	8	Second Data Byte
DQ[20:16]	5	ECC bits associated with first Data Byte
DQ[23:21]	3	3'b000
DQ[28:24]	5	ECC bits associated with second Data Byte
DQ[31:29]	3	3'b000

10.8.1 ECC Error Behavior

For correctable ECC errors, there is no error actively signaled via an interrupt or AXI response.

For uncorrectable ECC errors, the controller returns a SLVERR response back to the re-requesting AXI bus master. In both cases, information regarding the error (such as column, row and bank error address, error byte lane etc.,) is logged in the controller register space.

When the controller detects a correctable ECC error, it does the following:

- Sends the corrected data to the core as part of the read data.
- Sends the ECC error information to the register interface for logging.
- Performs a RMW operation to correct the data present in the DRAM (only if ECC scrubbing is enabled (`reg_ddrc_dis_scrub = 0`). This RMW operation is invisible to the core. Only one scrub RMW command can be outstanding in the controller at any time. No scrub is performed on single-bit ECC errors that occur while the controller is processing another scrub RMW.

When the controller detects an uncorrectable error, it does the following:

- Sends the uncorrectable data with an error response to the core. This results in an AXI SLVERR response on the AXI interface along with the corrupted data.
- Sends the ECC error information to the register module for logging.

10.8.2 Data Mask During ECC Mode

ECC is calculated over a byte of data and hence any data byte can be masked if necessary with ECC enabled. This alleviates the need for the controller to perform a RMW operation when byte masking occurs.

Operating Modes

The operating mode register bits, `ddrc_reg_operating_mode`, can be polled to determine the current mode of operation of the controller. The different modes are:

- 000 – uninitialized. The controller might be in soft reset, or it might be out of soft reset, but DRAM initialization sequence has not yet completed.
- 001 – normal operating mode. The controller is ready to accept read and write requests and the controller can issue reads and writes to DRAM.
- 010 – DRAM is in power down mode.
- 011 – DRAM is in self refresh mode.
- 100 : 111 – For LPDDR2 designs only, indicates DRAM is in deep power down.

10.9 Programming Model

10.9.1 Changing Clock Frequencies

The process of changing clock frequencies is as follows:

1. Request the controller to place the DRAM into self refresh mode, by asserting `reg_ddrc_selfref_en`.
2. Wait until `ddrc_reg_operating_mode[1:0] == 11` indicating that the controller is in self refresh mode.
3. Change the clock frequency to the controller (see [DDR Clock Initialization](#)).
4. Update any registers which might be required to change for the new frequency. This includes static and dynamic registers. If the updated registers involve any of `reg_ddrc_mr`, `reg_ddrc_emr`, `reg_ddrc_emr2` or `reg_ddrc_emr3`, then go to step 5. Otherwise go to step 6.
5. Assert `reg_ddrc_soft_rstb` to reset the controller. When the controller is taken out of reset, it re-initializes the DRAM. During initialization, the mode register values updated in step 4 are written to DRAM. Anytime after de-asserting reset, go to step 6.
6. Take the controller out of self refresh by de-asserting `reg_ddrc_selfref_en`.

Note: This sequence can be followed in general for changing DDRC settings, in addition to just clock frequencies.

10.9.2 ECC Programming

The following details the ECC programming requirements. Note that these configurations are on top of the regular DDR initialization programming. Also note that initialization of the whole DDR space before reading any data from it is recommended, to prevent ECC error generation as a result of accessing uninitialized areas of memory. Refer to [ECC Initialization](#) section for further details.

Enabling ECC operation (Switching from Non-ECC Mode to ECC Mode)

1. Program reg_ddrc_soft_rstb to 0 (resets the controller)
2. Program the ECC mode by programming reg_ddrc_ecc_mode to 3'b100
3. Program reg_ddrc_dis_scrub to 1'b0
4. Program reg_ddrc_data_bus_width to 2'b0
5. Program reg_ddrc_soft_rstb to 1 (takes the controller out of reset)

Disabling the ECC Operation(Switching from ECC Mode to Non-ECC Mode)

1. Program the reg_ddrc_soft_rstb to 0 (resets the controller)
2. Program the ECC mode by programming the reg_ddrc_ecc_mode to 3'b000
3. Program the reg_ddrc_dis_scrub to 1'b1
4. Program the reg_ddrc_data_bus_width to 2'b00
5. Program the reg_ddrc_soft_rstb to 1 (takes the controller out of reset)

Monitoring ECC Status

1. CHE_CORR_ECC_ADDR_REG_OFFSET gives the bank/row/column information of the ECC error correction
2. CHE_UNCORR_ECC_ADDR_REG_OFFSET gives the bank/row/column information of the ECC unrecoverable error
3. B[0] of CHE_CORR_ECC_LOG_REG_OFFSET indicates correctable ECC status
4. B[0] of CHE_UNCORR_ECC_LOG_REG_OFFSET indicates uncorrectable ECC status
5. CHE_ECC_STATS_REG_OFFSET
 - B[7:0] -> gives the number of uncorrectable errors
 - B[15:8] -> gives the number of correctable errors

10.9.3 Power Down

Enable power down mode in the Master Control register. Once enabled, the DDRC automatically puts the DRAM into pre-charge all power down after the programmed number of idle cycles (DDRC_param_reg1.reg_ddrc_powerdown_to_x32).

A refresh request brings the DRAM out of power down. It goes back into power down after the idle period.

Any transaction brings the DRAM out of power down automatically.

Clearing the power down enable bit also brings the DRAM out of power down.

10.9.4 Self Refresh

Set the Self Refresh Request bit in the Master Control register. The DDRC puts the DRAM into self refresh as soon as the transaction buffers are empty.

Software must ensure that no transactions arrive. If transactions keep arriving the DDRC never puts the DRAM into self refresh.

The first valid transaction brings the DRAM out of self refresh.

10.9.5 Deep Power Down

Note: Deep power down only applies to LPDDR2 mode.

Set `reg_ddrc_deeppowerdown_en=1`. The DDRC puts the DRAM into deep power down as soon as the transaction buffers are empty. If transactions keep arriving the DDRC never puts the DRAM into deep power down.

`reg_ddrc_deep_powerdown_en` must be reset to 0 to take DRAM out of deep power down mode. During deep power down exit, the controller performs automatic DRAM initialization.

In LPDDR2, once `reg_ddrc_deep_powerdown_en` is reset to 0, there is a wait period (determined by register `reg_ddrc_deeppowerdown_to_x1024`) before the DRAM comes out of deep power down. The value from the spec for this register is 500 us.

Note that any command that comes in while the DRAM is in deep power down mode is stored in the CAM and is processed after deep power down exit and DRAM re-initialization.

Static Memory Controller

11.1 Introduction

The static memory controller (SMC) can be used either as a NAND flash controller or a parallel port memory controller supporting the following memory types:

- NAND flash
- Asynchronous SRAM
- NOR flash

System bus masters can access the SMC controller as shown in [Figure 11-1](#). The operational registers of the SMC are configured through an APB interface. The memory mapping for the SMC is described in [Chapter 4, System Addresses](#). The SMC handles all commands, addresses, data, and the memory device protocols. It allows the users to access the controller by reading or writing into the operational registers. The SMC is based on ARM's PL353 static memory controller.

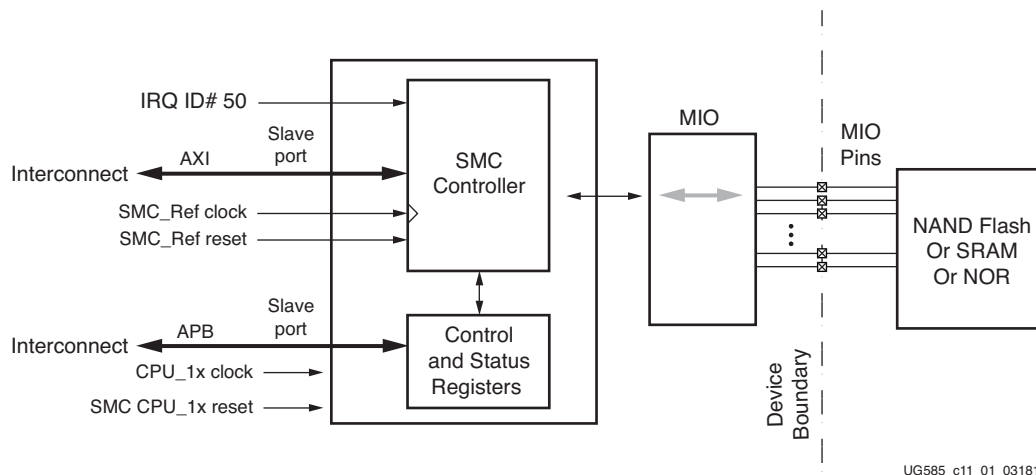


Figure 11-1: SMC System Level Diagram

11.1.1 Features

Features of the SMC are listed for each type of memory. The controller is configured to operate in one of two interface modes.

NAND Flash Interface

- *ONFI Specification 1.0*
- 8/16-bit IO width with a single chip select
- 16-word read and 16-word write data FIFOs
- 8-word command FIFO
- Programmable IO cycle timing
- 1-bit ECC hardware with software assist
- Asynchronous memory operating mode

Parallel (SRAM/NOR) Interface

- 8-bit data bus width
- One chip select with up to 25 address signals (64 MB)
- Two chip selects with up to 24 address (32 + 32 MB)
- 8-bit data width with up to 25 address signals
- Two chip select signals (with 24 address signals)
- 16-word read and 16-word write data FIFOs
- 8-word command FIFO
- Programmable I/O cycle timing on a per chip select basis
- Asynchronous memory operating mode

11.1.2 Block Diagram

The block diagram for the SMC is shown in Figure 11-2.

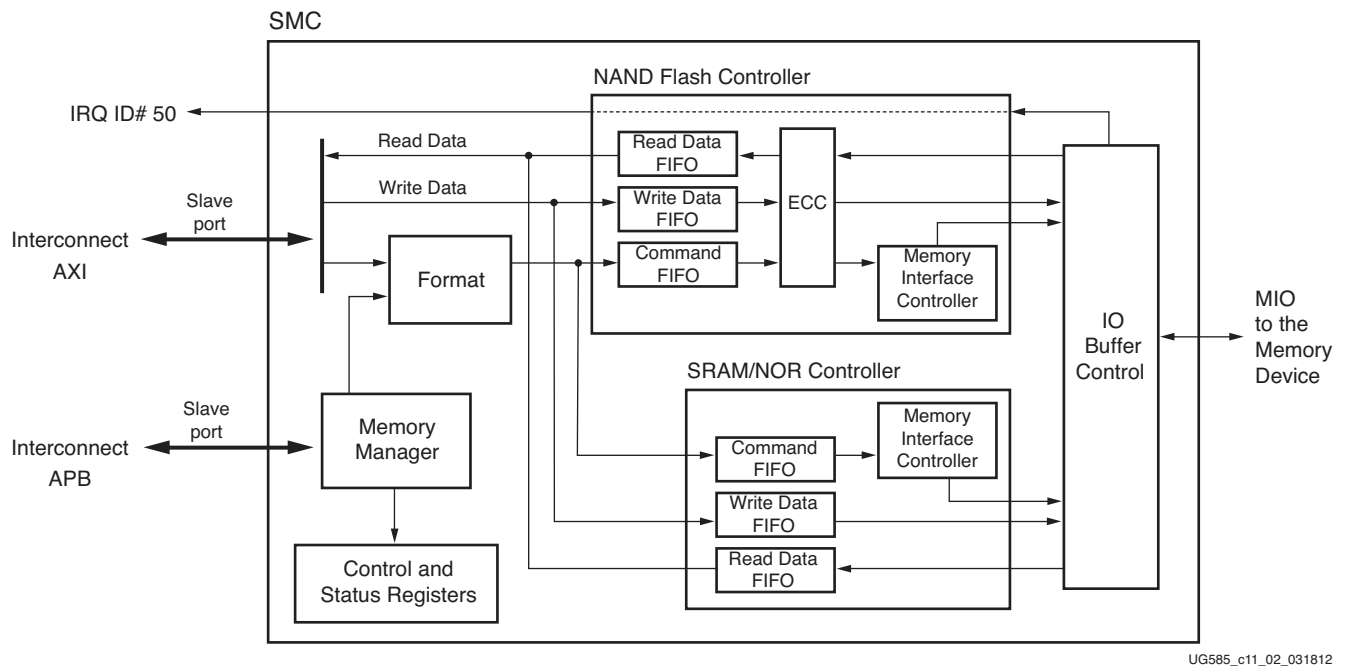


Figure 11-2: SMC Block Diagram

Interconnect Interfaces

For the NOR/SRAM controller mode, the AXI interface is memory mapped so software can read and write to/from memory. For the NAND flash controller mode, software writes commands to the NAND controller via the AXI interface. Details can be found in the ARM specification.

The APB bus interface provides a memory mapped area for the software to read and write the control and status registers.

Memory Manager

The memory manager tracks and controls the current state of the CPU_1x clock domain state machine. This block is responsible for updating register values that are used in the memory clock domain and controlling direct commands issued to memory and controlling entry-to and exit-from low-power mode through the APB interface.

Format

The format block arbitrates between memory accesses from the AXI slave interface and the memory manager. Requests from the manager have the highest priority. Requests from AR and AW channels are arbitrated on a round-robin basis. The format block also maps AXI transfers onto appropriate memory transfers and passes these to the memory interface through the command FIFO.

11.1.3 Notices

7z010 CLG225 Device

The 7z010 CLG225 device does not support the NOR/SRAM interface.

The NAND interface is supported in the 8-bit interface, but not the 16-bit interface.

11.2 Functional Operation

The functional operation of the SMC is described in the *ARM Static Memory Controller (PL350 series) Technical Reference Manual*. Additional information is provided in the following sections.

11.2.1 Boot Device

The NOR and NAND Flash controllers can be configured as a boot device. Its memory interface can only be routed through the MIO.

11.2.2 Clocks

The SMC has two clock domains that are driven by the CPU_1x and SMC_Ref clocks, see [Table 11-1](#). These clocks are controlled by the clock generator, refer to [Chapter 25, Clocks](#). The two clock domains are asynchronous to each other. The main benefit of asynchronous clocking is to maximize the memory performance while running the interconnect interface at a fixed system frequency. Additionally, in sleep-mode situations (when the system is not required to do much work) the frequency is lowered to reduce power consumption.

Table 11-1: SMC Clocks and Resets

Clock	Resets	Clock Domain	Description
CPU_1x	CPU_1x	Interconnect domain	This clock runs at 1/6 th or 1/4 th the CPU clock rate depending on the CPU clock mode. To stop this clock, first put the SMC in low-power mode.
SMC_Ref	SMC_Ref	SMC domain	This clock is used to control the I/O memory interfaces.

11.2.3 Resets

The controller has two reset inputs that are controlled by the reset subsystem; refer to [Chapter 26, Reset System](#). This SMC CPU_1x reset is used for the AXI and APB interfaces. The SMC_Ref reset is for the FIFOs and the rest of the controller including the control and status registers.

11.2.4 Interrupts

A single interrupt is generated by the controller, IRQ ID # 50. The interrupt is triggered on the rising edge of the busy input for the NAND memory interface. The SRAM/NOR interface does not generate an interrupt. The cause of the SMC reset is determined by reading the relevant status registers.

11.2.5 PL353 Functionality

The SMC is based on ARM's PL353 Primecell core and is hard-coded such that controller 0 can operate in SRAM/NOR mode and controller 1 can operate in NAND flash mode. The SRAM/NOR or NAND interface can be used in a system, but not both. The SRAM/NOR interface does not support PSRAM. The NAND flash controller does not support wear leveling.

When referencing ARM documentation, for programming and other purposes, refer to the implementation notes in [Table 11-2](#).

Table 11-2: SMC PL353 Implementation Notes

Parameter	Value	Design Notes
Chip Selects (Interface 0)	2	SRAM/NOR interface chip selects operate independently.
Chip Select (Interface 1)	1	NAND flash interface chip select
NAND flash mode data width	16	Data width can be 8 or 16 bits
SRAM mode data width	8	Data width is 8 bits.
System interface bus width	32	AXI
System interface clock rate	~	CPU_1x (1/6 th or 1/4 th the CPU clock frequency)
Command FIFO depth	8	Maximum supported depth on both interfaces
Read data word FIFO depth	16	Maximum supported depth on both interfaces
Write data word FIFO depth	16	Maximum supported depth on both interfaces
ECC support	Yes	1-bit ECC hardware with assistance from software
ECC Extra Block	Yes	Supported

11.2.6 Address Map

The registers and memory base address are listed in [Table 11-3](#).

Table 11-3: SMC Address Map Summary

Base Address	Mnemonic	Description	Type
0xE000_E000	SMC	Configuration registers base address	Registers
0xE100_0000	SMC_NAND	SMC NAND memory base address	Memory
0xE200_0000	SMC_SRAM0	SMC SRAM Chip Select 0 base address	Memory
0xE400_0000	SMC_SRAM1	SMC SRAM Chip Select 1 base address	Memory

11.3 I/O Signals

The MIO pin assignments for SRAM/NOR and NAND flash connections are shown in Table 11-4. The SMC interface signals are routed only to the MIO pins, they are not available on the EMIO interface. The MIO pins and restrictions (no NOR/SRAM and only 8-bit NAND) are shown in the MIO table in section 2.4.4 MIO-at-a-Glance Table.

Table 11-4: SMC MIO Pins

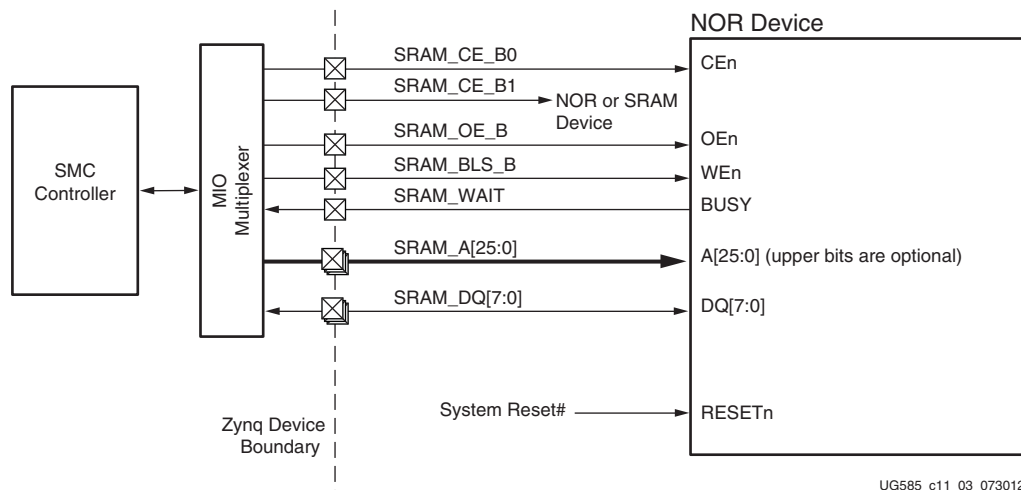
MIO Pin	SRAM/NOR Interface Mode				MIO Pin	NAND Flash Interface Mode			
	Signal Name	I/O	Default Value	Description		Signal Name	I/O	Default Value	Description
MIO Voltage Bank 0									
0	SRAM_CE_B[0]	O	-	SRAM/NOR chip sel 0	0	NAND_CE_B	O	-	NAND chip select
1	SRAM_CE_B[1]	O	-	SRAM/NOR chip sel 1	1	-	-	-	-
2	-	-	-	-	2	NAND_ALE	O	-	NAND address latch
3	SRAM_DQ[0]	IO	0	SRAM/NOR data	3	NAND_WE_B	O	-	NAND write enable
4	SRAM_DQ[1]	IO	0	SRAM/NOR data	4	NAND_IO[2]	IO	0	NAND data/address/cmd
5	SRAM_DQ[2]	IO	0	SRAM/NOR data	5	NAND_IO[0]	IO	0	NAND data/address/cmd
6	SRAM_DQ[3]	IO	0	SRAM/NOR data	6	NAND_IO[1]	IO	0	NAND data/address/cmd
7	SRAM_OE_B	O	-	SRAM/NOR output en	7	NAND_CLE	O	-	NAND chip select
8	SRAM_BLS_B	O	-	SRAM/NOR write en	8	NAND_RE_B	O	-	NAND read enable
9	SRAM_DQ[6]	IO	0	SRAM/NOR data	9	NAND_IO[4]	IO	0	NAND data/address/cmd
10	SRAM_DQ[7]	IO	0	SRAM/NOR data	10	NAND_IO[5]	IO	0	NAND data/address/cmd
11	SRAM_DQ[4]	IO	0	SRAM/NOR data	11	NAND_IO[6]	IO	0	NAND data/address/cmd
12	SRAM_WAIT	I	-	SRAM/NOR wait	12	NAND_IO[7]	IO	0	NAND data/address/cmd
13	SRAM_DQ[5]	IO	0	SRAM/NOR data	13	NAND_IO[3]	IO	0	NAND data/address/cmd
14	-	-	-	-	14	NAND_BUSY	I	0	NAND busy
15	SRAM_A[0]	O	-	SRAM/NOR address	15	-	-	-	-
MIO Voltage Bank 1									
23:16	SRAM_A [8:1]	O	-	SRAM/NOR address	23:16	NAND_IO [15:8]	IO	0	NAND data/address/cmd
39:24	SRAM_A [24:9]	O	-	SRAM/NOR address	39:24	-	-	-	-

Optional Pins

- The SRAM/NOR Busy signal is not used when an SRAM is connected.
- For either SRAM or NOR, the upper address bits are optional. When not used, they can be assigned to other functions.

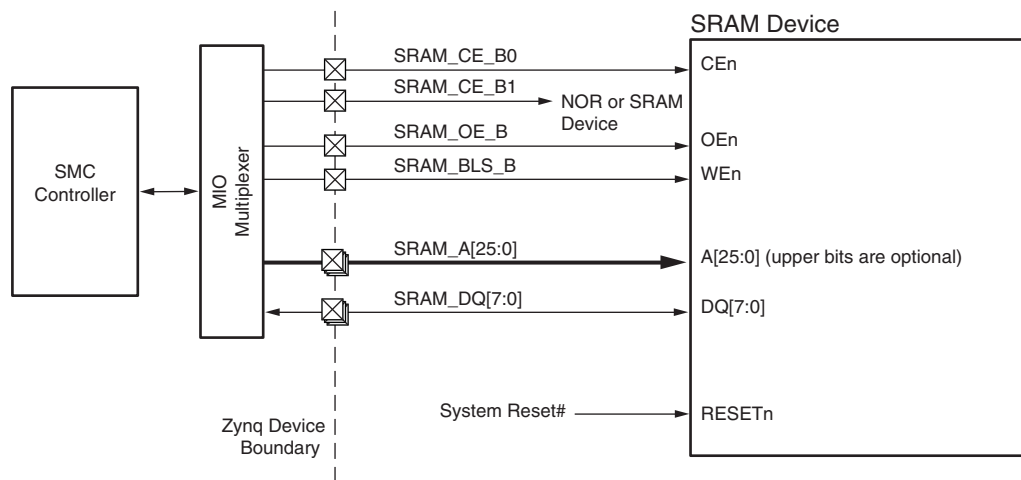
11.4 Wiring Diagrams

The SMC supports the configurations shown in Figure 11-3, Figure 11-4, and Figure 11-5. The NOR/SRAM mode of the SMC can support two devices (NOR and/or SRAM) using chip selects 0 and 1.



UG585_c11_03_073012

Figure 11-3: NOR Device Wiring Diagram



UG585_c11_04_073012

Figure 11-4: SRAM Device Wiring Diagram

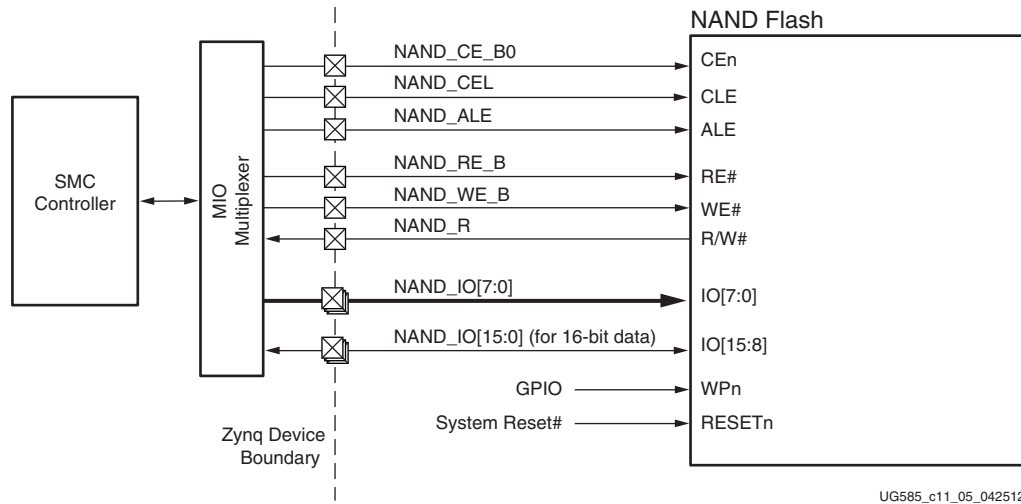


Figure 11-5: NAND Flash Device Wiring Diagram

11.5 Register Overview

The SMC registers are summarized in [Table 11-5](#).

Table 11-5: SMC Register Overview

Controller	Register Name	Description
Both	MEMC_STATUS	Operating and interrupt status, read-only
	MEMIF_CFG	SMC configuration information, read-only
	MEMC_CFG_{SET, CLR}	Enable/disable/clear interrupts and control low power state
	DIRECT_CMD	Issue a set command, write-only
	SET_{CYCLES, OPMODE}	Stage a cycles or opmode operation to the SRAM/NOR and NAND flash registers
	USER_{STATUS, CONFIG}	
SRAM/NOR CS 0, 1	REFRESH_PERIOD_{0,1}	Insert idle cycles between SRAM/NOR burst cycles
	SRAM_CYCLES0_{0,1}	Timing cycles
	OPMODE0_{0,1}	Operating mode
NAND Flash	NAND_CYCLES1_0	Timing cycles
	OPMODE1_0	Operating mode
	ECC_{STATUS, MEMCFG}_1	ECC status and configuration
	ECC_MEMCOMMAND{2:1}_1	Commands used for ECC reads and writes
	ECC_ADDR{1:0}_1	Address generated by controller
	ECC_VALUE{3:0}_1	Value generated by controller

11.6 Programming Model

The programming model is described in the *ARM Static Memory Controller (PL350 series) Technical Reference Manual* (see [Appendix A, Additional Resources](#)). The configuration of the SMC is summarized in [Table 11-2](#).

Quad-SPI Flash Controller

12.1 Introduction

The Quad-SPI flash controller is part of the input/output peripherals (IOP) located within the PS. It is used to access multi-bit serial flash memory devices for high throughput and low pin count applications.

The controller operates in one of two modes: I/O mode or linear addressing mode. In I/O mode, software interacts closely with the flash device protocol. The software writes the flash commands and data to the controller using the four TXD registers. Software reads a register that contains the data received from the flash device. The I/O mode supports all memory device operations, such as program, erase, and read.

Linear addressing mode eliminates the software overhead that the I/O mode requires to read and write the flash memory. Linear Mode engages hardware to issue commands to the flash memory and control the flow of data from the flash memory bus to the AXI interface. The controller responds to memory requests on the AXI interface as if the flash memory were a ROM memory.

The controller can interface to one or two flash devices. Two devices can be connected in parallel for 8-bit performance, or in a stacked, 4-bit arrangement to minimize pin count. The two device combinations are shown in [Figure 12-1](#).

12.1.1 Features

- 32-bit AXI interface for Linear Addressing Mode and I/O transfers
- 32-bit register programming interface via APB
- Programmable bus protocol for flash memories from Micron, Winbond, and Spansion
- Legacy SPI and scalable performance: 1x, 2x, 4x, 8x I/O widths
- Flexible I/O
 - Single SS 4-bit I/O flash interface mode
 - Dual SS 8-bit separate parallel I/O flash interface mode
 - Dual SS 4-bit shared stacked I/O flash interface mode
 - Single SS, legacy SPI interface
- 16 MB addressing per device (32 MB for two devices)
- Device densities up to 128 Mb

- I/O mode (flash commands and data)
 - Software issues instructions and manages flash operations
 - Interrupts for FIFO control
 - 256-byte RxFIFO, 256-byte TxFIFO
- Linear addressing mode (executable read accesses)
 - Memory reads and writes are interpreted by the controller
 - AXI port buffers up to four read requests
 - AXI incrementing and wrapping address functions

12.1.2 System Viewpoint

The Quad-SPI flash controller is part of the IOP and connects to external SPI flash memory through the MIO as shown in Figure 12-1. The controller supports one or two memories.

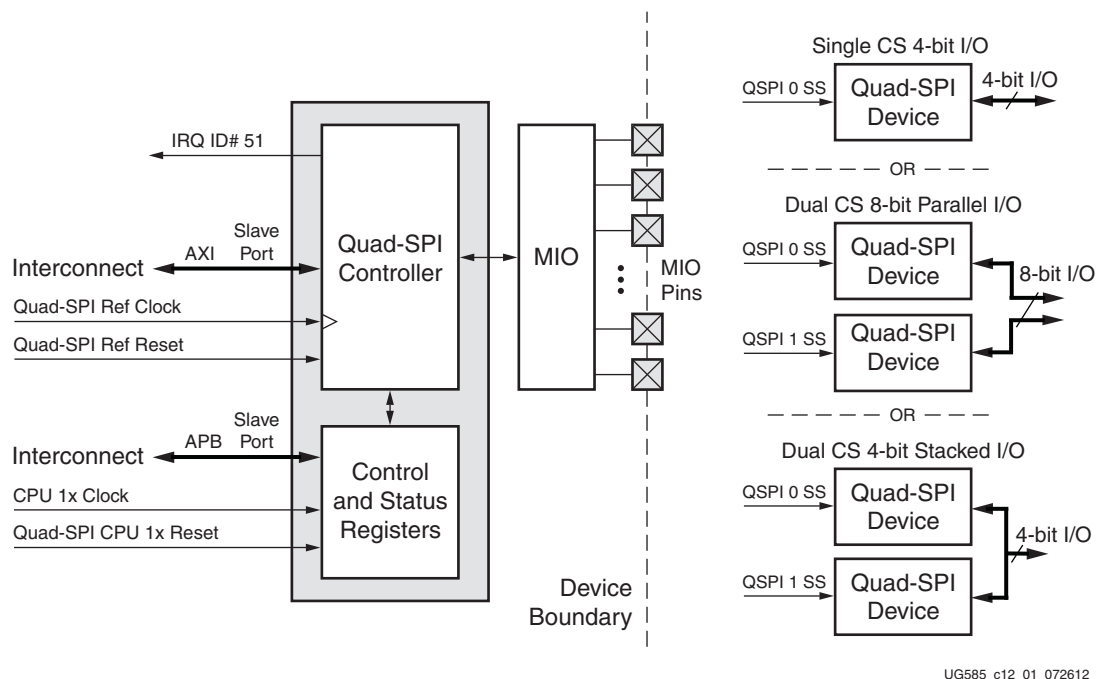


Figure 12-1: Quad-SPI Controller System Viewpoint

Address Map and Device Matching

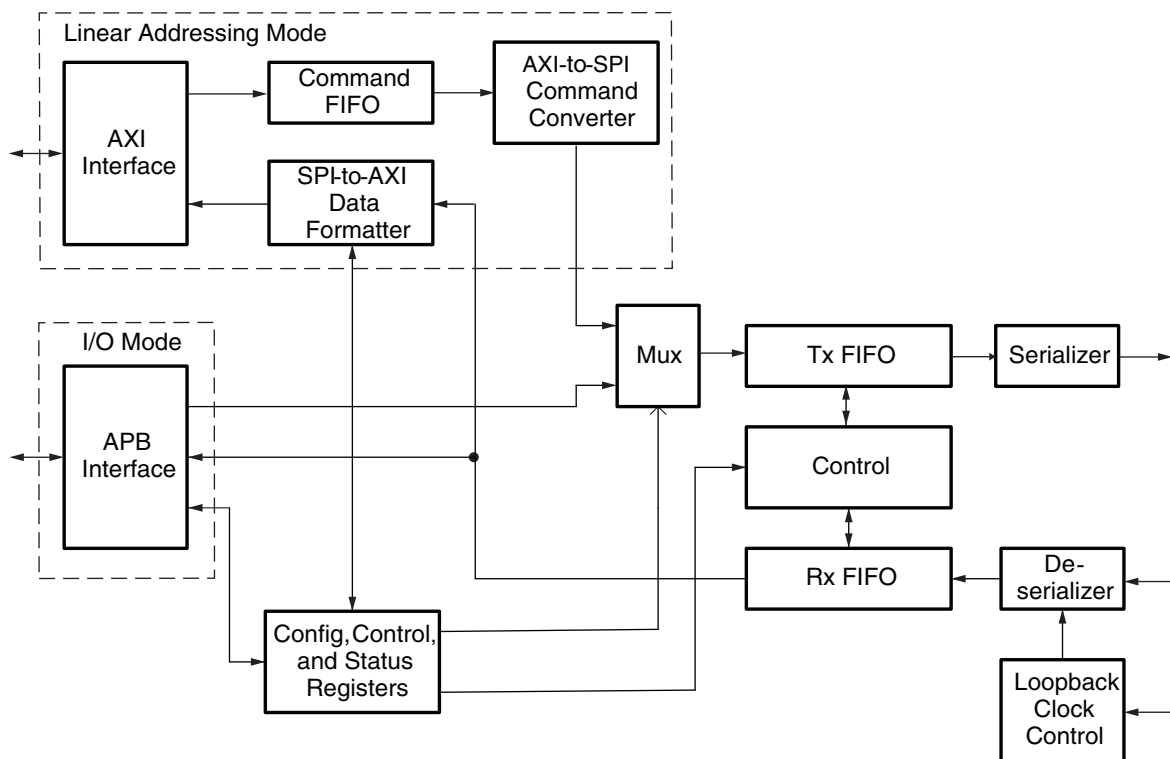
When a single device is used, the address map starts at `FC00_0000` and goes to a maximum of `FCFF_FFFF` (16 MB). The address map for a two-device system depends on the memory devices and the I/O configuration. In two-device systems, the Quad-SPI devices need to be from the same vendor so they have the same protocol.

The 8-bit parallel I/O configuration also requires that the devices have the same capacity. The address map for the parallel I/O configuration starts at `FC00_0000` and goes to the address of the combined memory capacities, up to a maximum of `FDFE_FFFF` (32 MB).

For the 4-bit Stacked I/O configuration, the devices can have different capacities, but must have the same protocol. In this mode, the QSPI 0 device starts at `FC00_0000` and goes to a maximum of `FCFF_FFFF` (16 MB). The QSPI 1 device starts at `FD00_0000` and goes to a maximum of `FDFE_FFFF` (another 16 MB). If the first device is less than 16 MB in size, then there will be a memory space hole between the two devices.

12.1.3 Block Diagram

The block diagram of the is shown in [Figure 12-2](#).



UG585_c12_02_071712

Figure 12-2: Quad-SPI Controller Block Diagram

12.1.4 Notices

Operating Restrictions

When a single device is used, it must be connected to QSPI 0. When two devices are used, both devices must be identical (same vendor and same protocol sequencing).

The MIO pins for the Quad-SPI controller conflict with both the NOR and NAND interfaces of the SMC controller. The NOR/SRAM and NAND interfaces cannot be used when Quad-SPI is used. More information about the MIO pins is provided in section [2.4 MIO-EMIO](#).

12.2 Functional Description

The Quad-SPI flash controller can operate in either I/O mode or linear addressing mode. For reads, the controller supports single, dual and quad read modes in both I/O and linear addressing modes. For writes, single and quad modes are supported in I/O mode. Writes are not supported in linear addressing mode.

12.2.1 Operational Modes

Quad-SPI operating mode transitions are shown in Figure 12-3.

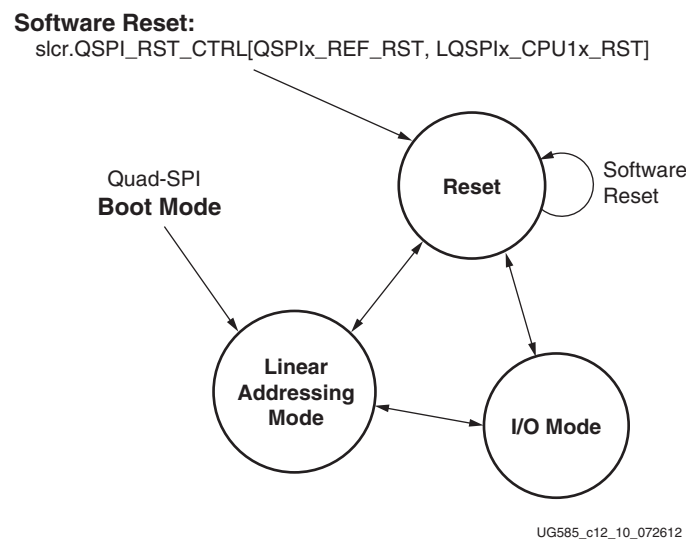


Figure 12-3: Quad-SPI Operating Mode Transitions

In I/O mode, software is responsible for almost all aspects of read data management. In linear mode, the controller carries out all necessary read data management and the memory reads like a ROM to software.

12.2.2 I/O Mode

In I/O mode, the software is responsible for preparing and formatting commands and data into instructions according to the Quad-SPI protocol. Instructions for the Quad-SPI device consist of a sequence of one or more 4-byte words. The instructions are written to the TxFIFO via the APB interface. The transmit logic serializes the instruction and sends it out to the flash memory.

The controller handles one write at a time and it is important that the software uses status bits to wait until the previous instruction has completed before sending in a new instruction, or incorrect operation will result. While the controller is sending an instruction, it concurrently samples the raw serial read data signal, performs serial-to-parallel conversion, and stores the data in the RxFIFO.

Software needs to filter the raw data from the RxFIFO to obtain the relevant data content. The controller does not modify either the instruction written by software or the captured data put into the RxFIFO.

The controller supports little endian mode and the most significant bit of the least significant byte of a 4-byte word of an instruction is sent first.

Flow Control

I/O mode uses two modes of flow control during data transfers: manual mode and automatic mode. In manual mode, chip select is asserted manually by software before the data transfer and is de-asserted by software after the data transfer completion. In automatic mode, chip select assertion/de-assertion is done by the controller automatically without software intervention.

Manual mode of flow control is done using manual start, which starts the data transfer from the TxFIFO. Chip select assertion should occur before the setting the manual start bit. After the data is transferred from the TxFIFO, chip select can still remain asserted to continue the data transfer. Software shall write new data into TxFIFO and assert manual start to start the data transfer. After all the data is transferred, chip select shall be de-asserted indicating the end of transfer.

Using manual mode of flow control, software can transfer data greater than the RxFIFO size, with single chip select assertion/de-assertion sequence, and the corresponding command for the entire data should be sent only at the start of transfer.

Automatic mode of flow control starts transferring data from the moment the command/data is available in the TxFIFO. the Quad-SPI controller asserts chip select before the start of data transfer and de-asserts at the end of data transfer automatically. To start a new transfer, software needs to fill data in the TxFIFO again. Using automatic mode of flow control, software should fill in all the data that should be sent to the flash in a chip select assertion and de-assertion cycle, into the TxFIFO. Maximum data that can be sent is limited by the TxFIFO depth, which is 256 bytes.

FIFO Reads and Writes

Reading from RxFIFO:

- To receive "n" bytes of data in the RxFIFO from the Quad-SPI flash device, transfer "n" dummy bytes from the TxFIFO over the data lines.
- Wait until the transfer of "n" dummy bytes from TxFIFO is completed. This will mark the end of data reception in RxFIFO.

Writing to TxFIFO:

- For every "n" bytes transmitted from TxFIFO over the data lines to Quad-SPI flash, there will be "n" bytes stored in RxFIFO.
- The user needs to flush the "n" bytes by reading the RXFIFO before starting the next transfer.

Transmit Registers (TXD)

Software writes byte sequences that are needed for the specific flash device. Refer to the vendor's specification. The controller has four write-only 32-bit TXD registers for software to issue a byte

stream of commands to get status and read/write data from the flash memory. Quad-SPI TXD register write formats are described in [Table 12-1](#).

Table 12-1: Quad-SPI TXD Register Write Formats

Register Name	Bit Field				Example Usage
	31:24	23:16	15:8	7:0	
TXD 1	Reserved	Reserved	Reserved	Command only	Set write enable
TXD 2	Reserved	Reserved	Data 0	Command only	Write status with data
TXD 3	Reserved	Data 1	Data 0	Command only	Read status with two dummy bytes
TXD 0	Data 3	Data 2	Data 1	Data 0	Write Data to Transmit or Dummy data for Reads

12.2.3 Linear Addressing Mode

The controller has a 32-bit AXI slave interface to support linear address mapping for read operations. When a master issues an AXI read command through this port, an internal state machine generates I/O commands to load the corresponding memory data and send it back through the AXI interface.

In linear mode, the flash memory subsystem behaves like a typical read-only memory with an AXI interface that supports a command pipeline depth of four. The controller is responsible for carrying out AXI-to-SPI protocol conversion. The linear mode improves both the user friendliness and the overall read memory throughput over that of the I/O mode by lessening the amount of software overhead that is required. From a software perspective, there is no perceived difference between accessing the linear Quad-SPI memory subsystem and that of other ROMs, except for a potentially longer latency.

A simplified block diagram of the controller showing the linear and I/O portions is shown in [Figure 12-2](#).

AXI Interface Operation

Only AXI read commands are supported by the linear addressing mode. All valid write addresses and write data are acknowledged immediately but are ignored, that is, no corresponding programming (write) of the flash memory is carried out. All AXI writes generate an SLVERR error on the write response channel.

Both incrementing- or wrapping-address burst reads are supported. Fixed-address burst is not supported and will cause an SLVERR error. Therefore, the only recognized arburst[1:0] value is either 2'b01 or 2'b10. All read accesses must be word-aligned and the data width must be 32-bits (no narrow burst transfers are allowed).

[Table 12-2](#) lists the read address channel signals from a master that are ignored by the interface.

Table 12-2: Ignored AXI Read Address Channel Signals

Signal	Value
araddr[1:0]	Ignored, assumed to be 0, i.e. always assumed to be word aligned
arsize[2:0]	Ignored, always a 32-bit interface
arlock[1:0]	Ignored
arcache[3:0]	Ignored
arprot[2:0]	Ignored

The AXI slave interface provides a read acceptance capability of 4 so that it can accept up to four outstanding AXI read commands.

Interface Configuration and Read Modes

AXI read burst transfers are translated into SPI flash read instructions that are sent to the controller's TxFIFO. The transmit logic retrieves the read instructions from the TxFIFO and passes them to the SPI flash memory device according to the SPI protocol.

Software defines the SPI read command that is used in linear addressing mode by writing to `qspi.LQSPI_CFG[INST_CODE]`. The supported read command codes and the recommended configuration register settings (`qspi.LQSPI_CFG`) are listed in [Table 12-3](#).

Table 12-3: Quad-SPI Device Configuration Register Values

Operating Mode	Instruction Code	Winbond & Spansion		Micron	
		1 Device	2 Devices	1 Device	2 Devices
Read (serial bit)	0x03	0x80000003	0xE0000003	0x80000003	0xE0000003
Fast Read (serial bit)	0x0B	0x8000010B	0xE000010B	0x8000010B	0xE000010B
Dual Output Fast Read	0x3B	0x8000013B	0xE000013B	0x8000013B	0xE000013B
Quad Output Fast Read	0x6B	0x8000016B	0xE000016B	0x8000016B	0xE000016B
Dual I/O Fast Read	0xBB	0x82FF00BB	0xE2FF00BB	0x82FF01BB	0xE2FF01BB
Quad I/O Fast Read	0xEB	0x82FF02EB	0xE2FF02EB	0x82FF04EB	0xE2FF06EB

Unsupported Devices

A number of devices implement custom 4-bit wide SPI-like interfaces for flash memory access, such as the SQI devices from SST, and the Fast4 devices from Atmel. Some other Quad-SPI devices, like some Micron/Numonyx devices, offer an option to switch operation to such a custom 4-bit interface, through a non-volatile configuration bit.

These interfaces operate differently from the devices supported by the Quad-SPI controller. These flash memory devices operate in 4-bit mode during the instruction phase, as well as the address and data phases. This requires the Quad-SPI flash controller to power up in 4-bit mode and remain in that mode permanently (or until configured otherwise, if that option is available). There are no plans to enable the support for these custom interfaces.

Quad IO mode (0xEB) for a two-memory shared bus is not supported.

Performance Modes

To get the highest performance, the user should use either the quad output mode or the quad IO mode.

Note that the Spansion device, while supported, is not be able to run at the full specified speed. Refer to the applicable Zynq-7000 EPP data sheet for operating frequencies.

Read Data Management

A 63-deep RxFIFO provides read data buffering to hold a minimum of three AXI burst transfers of 16 bytes each. Since the RxFIFO starts receiving data as soon as the chip-select signal is active, the linear address adapter removes incoming data that corresponds to the instruction code, if any, the address, and the dummy cycles.

The read data must be aligned with the corresponding word boundary specified by the address. For data alignment purposes, the controller can modify the address as illustrated in [Figure 12-4](#) before it is sent to the flash memory device. The address modification involves reducing the address by up to 3 byte locations such that the intended return data is word aligned automatically. The amount of address change is transparent to the AXI interface, and is instruction dependent.

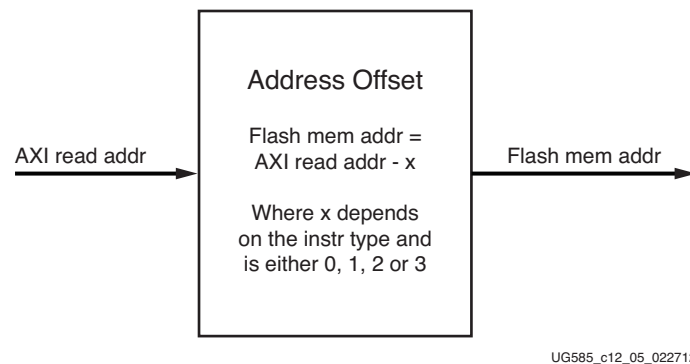


Figure 12-4: Automatic Address Offset For Word Alignment

Read Latency

In linear mode, the default read mode is fast quad I/O. For a single device, the number of clock cycles from the time an 8-bit instruction code and a 24-bit address is available to the time when the first 32-bit data becomes available is:

$$\begin{aligned}
 TI &= \text{instruction latency} + \text{address latency} + \text{overhead (including offset)} + \text{data latency} \\
 &= 8 \text{ cycles} + 6 \text{ cycles} + 10 \text{ cycles} + 8 \text{ cycles} \\
 &= 32 \text{ cycles}
 \end{aligned}$$

With the SPI clock of 100 MHz, the latency at the memory interface is 320 ns. Other read modes have higher latency and can be calculated in a similar manner.

12.3 Programming Guide

Example: Start-up Sequence

1. **Configure Clocks.** Refer to section [12.4.1 Clocks](#).
2. **Configure Tx/Rx Signals.** Refer to section [12.5.2 MIO Programming](#).
3. **Reset the Controller.** Refer to section [12.4.2 Resets](#).
4. **Configure the Controller.** Refer to section [12.3.1 Configuration](#).

Now, either configure the controller for linear addressing mode (section [12.2.3 Linear Addressing Mode](#)) or configure the controller for I/O mode (section [12.3.3 Configure I/O Mode](#) and section [12.3.4 I/O Mode Interrupts](#)).

12.3.1 Configuration

Example: Configure Controller

This example applies to both linear addressing and I/O modes. It prepares the controller baud rate, FIFO, flash mode, clock phase/polarity, and programs the loopback delay

The values to program into the `qspi.Config_reg` register are shown in [Table 12-3, page 259](#).

1. **Configure the controller.** Write to the `qspi.Config_reg` register.
 - a. Set baud rate, `[BAUD_RATE_DIV]`.
 - b. Select master mode, `[MODE_SEL] = 1`.
 - c. Select flash mode (not Legacy SPI), `[LEG_FLSH] = 1`.
 - d. Select Little Endian, `[endian] = 0`.
 - e. Set FIFO width to 32 bits, `[FIFO_WIDTH]`.
 - f. Set clock phase, `[CLK_PH]` and Polarity, `[CLK_POL]`.
2. **If baud rate divider is 2, then change default setting.** If the `qspi.Config_reg[BAUD_RATE_DIV]` is set to `0b00`, configure the `qspi.LPBK_DLY_ADJ` (loopback delay adjustment) register with the following settings:
 - a. **Set to select internal clock.** `qspi.LPBK_DLY_ADJ[USE_LPBK] = 1`.
 - b. **Set the clock delay 0.** `qspi.LPBK_DLY_ADJ[DLY0] = 0b00`.
 - c. **Set the clock delay 1.** `qspi.LPBK_DLY_ADJ[DLY1] = 0b00`.

12.3.2 Configure Linear Addressing Mode

Example: Linear Addressing Mode (Memory Reads)

The sequence of operations for data reads in linear addressing mode is as follows:

1. **Set manual start enable to auto mode.** Set `qspi.Config_reg[Man_start_en] = 0`.
2. **Assert the chip select.** Set `qspi.Config_reg[PCS] = 0`.
3. **Program the configuration register for linear addressing mode.** Example values are shown in [Table 12-3, page 259](#).
4. **Enable the controller.** Set `qspi.En_REG[SPI_EN] = 1`.
5. **Read data from the linear address memory region.** The memory range depends on the size and number of devices. The range is from `0xFC00_0000` up to `0xFDFE_FFFF`.
6. **Disable the controller.** Set `qspi.En_REG[SPI_EN] = 0`.
7. **De-assert chip select.** Set `QSPI.Config_reg[PCS] = 1`.

12.3.3 Configure I/O Mode

Example: I/O Mode (Memory Reads and Writes)

The sequence of operations uses I/O mode for reads and writes.

1. **Enable manual mode.** Write 1 to `qspi.Config_reg[Man_start_en, Manual_CS] = 1`.
2. **Configure dual flash mode,** refer to [Figure 12-6, page 271](#). Write to the `qspi.LQSPI_CFG[TWO_MEM, SEP_BUS]` register to dual flash mode.
3. **Assert chip select.** Set `qspi.Config_reg[PCS] = 0`.
4. **Enable the controller.** Set `qspi.En_REG[SPI_EN] = 1`.
5. **Write byte sequences to the flash memory.** Write from 1 to 4 bytes to the TxFIFO using the TXD registers.
6. **Avoid TxFIFO overflow.** When the TxFIFO is empty, 256 bytes can be written. After that, software can avoid overflowing the TxFIFO by reading `qspi.Intr_status_REG[TX_FIFO_full]` and waiting until it equals 0 before writing to a TXD register..
7. **Enable the interrupts.** Write to `qspi.Intrpt_en_REG`. Interrupt handlers that handle the interrupt conditions are discussed in interrupt handlers section.
8. **Start data transfer.** Set `qspi.Config_reg[Man_start_com] = 1`.
9. **Interrupt handler:** Transfer all the required data to QSPI flash during program/read operations to QSPI flash. (See [Example: I/O Mode Interrupt Service Routine, page 263](#).)
10. **If read operations are carried out:** re-arrange the READ data to eliminate the data read due to dummy cycles.
11. **Disable controller.** Set `qspi.En_REG[SPI_EN] = 0`.
12. **De-assert chip select.** Set `QSPI.Config_reg[PCS] = 1`.

Note that the TxFIFO width must be programmed to 32 bits: `qspi.Config_reg[FIFO_WIDTH] = 0b11`. Software needs to take care of “consecutive non word aligned” transfers.

Example: I/O Mode Interrupt Service Routine

1. **Configure the ISR to handle the interrupt conditions based on the Quad-SPI device type.** The simplest ISR reads data from the RxFIFO and writes content to the TxFIFO. The system interrupt controller (GIC) is described in [Chapter 7, Interrupts](#). The controller generates a system peripheral interrupt (SPI), IRQ ID #51. The interrupt mechanism for the Quad-SPI controller is described in section [12.3.4 I/O Mode Interrupts](#).
 - a. Read transfer interrupt. RxFIFO Not Empty Interrupt
 - b. Write transfer interrupt. TxFIFO Not Full Interrupt

Rx/Tx FIFO Response to I/O Command Sequences

Example command and sequences:

- Write Enable Command
- Read Status Command
- Read Data Sequence

In these examples, YY can have any value. Each YY pair could have a different value.

To receive data in serial legacy mode, the value is sampled from MISO/DQ1 line into RxFIFO synchronous to clock, while the command and address transactions happen on MOSI/DQ0.

Example: Write Enable Command (code 0x06)

1. **Send the Write Enable Command (WREN).** Write `0xYYYY_YY06` to the `qspi.TXD1` register.
 - a. WREN command = `0x06`.
 - b. YY = 0.
 - c. The controller shifts one bytes out of the TxFIFO to the device and receives two bytes in the RxFIFO.
2. **Read Status.** Reads the `qspi.RXD` register and receive `0xYYPP_PPPP`.
 - a. Value is `0x0000_0000` when YY = `0x0` (the status) and PP_PPPP = `0x0` (previous state of the bits).
 - b. Software remembers that one byte resulted from the Write Enable command and returns `0xYY` to the calling function.

The content in the RxFIFO after sending the WREN command follows. (Previous means that the value has not changed from the register's previous value.)

RxFIFO Entry	MSB			LSB
1	Invalid	Invalid	Invalid	Invalid
0	00	Previous	Previous	Previous

Example: Read Status Command (code 0x05)

1. **Send the Read Status Command (RDSR).** Write 0xYYYY_DD05 to the qspi.TXD2 register.
 - a. Command is 0x05, DD = dummy data, YY = 0
 - b. The controller shifts two bytes out of the TxFIFO to the flash memory and receives two bytes in the RxFIFO.
2. **Read Status Value.** Read 0xZZYY_PPPP from the qspi.RXD register.
 - a. Value is 0x0300_0000 when ZZ = 0x03, YY == 0x0 and PPPP = 0x0.
 - b. Software remembers that two bytes are valid and returns 0x00, 0x03 to the calling function.

The content in the RxFIFO after sending the RDSR command is shown in the table (prev means the value has not changed from the register's previous value):

RxFIFO Entry	MSB			LSB
1	Invalid	Invalid	Invalid	Invalid
0	03	0x00	Previous	Previous

Example: Read Data Sequence

This example returns the four bytes of data at address 0 to the calling function.

1. **Send the data read instruction.** Write 0xA2A1_A003 to the qspi.TXD0 register.
 - a. Instruction includes command (0x03) plus address (A0, A1 and A2).
2. **Send dummy data.** Write 0xD0D1_D2D3 (dummy data) to qspi.TXD0 register (second TxFIFO entry).
 - a. The controller shifts 8 bytes out of the TxFIFO to the flash memory and receives 8 bytes in the RxFIFO.

The content of the TxFIFO for this example follows. The byte sequence from controller to the device is: 0x03, Y0, Y1, Y2, D0, D1, D2 and D3.

RxFIFO Entry	MSB			LSB
1	D3	D2	D1	D0
0	A2	A1	A0	0x03

3. **Read past the instruction word.** Read the qspi.RXD register and receive 0xYYYY_YYYY:
 - a. YY = 0.
4. **Read flash memory data.** Read the RXD register again and receives 0xD3D2_D1D0.
 - a. For the second read, software remembers that four bytes are valid.
 - b. Example data: 0x2468ACEF.
 - c. Overall, software reads these bytes: 0x00, 0x00, 0x00, 0x00, 0x24, 0x68, 0xAC, 0xEF and returns the four bytes of data to the calling function.

The content of the RxFIFO for this example follows. The byte sequence from the device to controller is: YY, YY, YY, YY, 0xEF, 0xAC, 0x68 and 0x24.

RxFIFO Entry	MSB			LSB
1	0x24	0x68	0xAC	0xEF
0	YY	YY	YY	YY

12.3.4 I/O Mode Interrupts

Interrupts are only used in I/O mode. The controller interrupt is asserted whenever any of the interrupt conditions are met. The Quad-SPI interrupt handler checks the cause of the interrupt. A single interrupt service routine can manage all of the interrupt conditions.

Example: Interrupt Handler for Rx and Tx

The interrupt handler is trigger by IRQ ID #51. The example reads the RxFIFO until it is empty and then fills-up the TxFIFO. The RxFIFO Not Empty Interrupt status is used to determine if content can be read from the RxFIFO. The TxFIFO Not Full interrupt indicates if there is room in the TxFIFO for more content.

1. **Disable all of the interrupts in the controller.** Set `qspi.Intrpt_dis_REG = 0x0`.
2. **Clear the interrupts.** Read the interrupt status register `qspi.Intr_status_REG`.
3. **Empty the RxFIFO.** Check if RxFIFO Not Empty interrupt is asserted. If `qspi.Intr_status_REG[RX_FIFO_not_empty] = 1`, then there is data in the RxFIFO.
 - a. **If the status is asserted, then read data from the RxFIFO.** Read the data using the `qspi.RX_data_REG` register.
 - b. **Read data from the RxFIFO and poll the interrupt status until the RxFIFO is empty.** The RxFIFO is empty when `qspi.Intr_status_REG[RX_FIFO_not_empty] = 0`.
4. **Fill the TxFIFO.** Check if the TxFIFO Not Full status is asserted. If `qspi.Intr_status_REG[TX_FIFO_not_Full] = 1`, then there is data to be sent to the flash device (program and/or read operations):
 - a. Write data to the `qspi.TXD0` register.
 - b. Poll for `qspi.Intr_status_REG[TX_FIFO_full] = 1`, which indicates TX FIFO is full.
 - c. Follow steps a,b till all the data is written to the TxFIFO or until `qspi.Intr_status_REG[TX_FIFO_full] = 1`.
5. **Enable the interrupts.** Set `qspi.Intrpt_en_REG[TX_FIFO_not_full, RX_FIFO_full]` both = 1.
6. **Start the data transfer.** Set `qspi.Config_reg[MANSTRTEN] = 1`.

12.3.5 Rx/Tx FIFO Response to I/O Command Sequences

Example command and sequences:

- Write enable command
- Read status command

- Read data sequence

In these examples, YY can have any value. Each YY pair could have a different value.

To receive data in serial legacy mode, the value is sampled from MISO/DQ1 line into RxFIFO synchronous to clock, while the command and address transactions happen on MOSI/DQ0. The other I/O configuration modes have different formats.

Example: Write Enable Command (code 0x06)

1. **Send the Write Enable Command (WREN).** Write 0xYYYY_YY06 to the qspi.TXD1 register.
 - a. WREN command = 0x06.
 - b. YY = 0.
 - c. The controller shifts one bytes out of the TxFIFO to the device and receives two bytes in the RxFIFO.
2. **Read Status.** Reads the qspi.RXD register and receive 0xYYPP_PPPP.
 - a. Value is 0x0000_0000 when YY = 0x00 (the status in this example, but it could be another value) and PP_PPPP = 0x0 (previous state of the bits).
 - b. Software remembers that one byte resulted from the Write Enable command and reads 0xYY.

The content in the RxFIFO that software is expected to read after sending the WREN command is as follows:

RxFIFO Register Read	MSB			LSB
RXFIFO_DATA1	00	sw don't care	sw don't care	sw don't care

Example: Read Status Command (code 0x05)

1. **Send the Read Status Command (RDSR).** Write 0xYYYY_DD05 to the qspi.TXD2 register.
 - a. Command is 0x05, DD = dummy data, YY = 0 (example)
 - b. The controller shifts two bytes out of the TxFIFO to the flash memory and receives two bytes in the RxFIFO.
2. **Read Status Value.** Read 0xZZYY_PPPP from the qspi.RXD register.
 - a. Value is 0x0300_0000 when ZZ = 0x03, YY == 0x0 and PPPP = 0x0.
 - b. Software remembers that two bytes are valid and reads 0x00, 0x03.

The content in the RxFIFO after sending the RDSR command is as follows:

RxFIFO Register Read	RxFIFO Entry	MSB			LSB
RXFIFO_DATA1	0	03	0x00	sw don't care	sw don't care

Example: Read Data Sequence

In this example, software reads the four bytes of data at address 0.

1. **Send the Data Read Instruction.** Write `0xA2A1_A003` to the `qspi.TXD0` register.
 - a. Instruction includes command (`0x03`) plus address (A0, A1 and A2).
2. **Send Dummy Data.** Write `0xD0D1_D2D3` (dummy data) to `qspi.TXD0` register (second TxFIFO entry).
 - a. The controller shifts 8 bytes out of the TxFIFO to the flash memory and receives 8 bytes in the RxFIFO.

The content of the TxFIFO for this example follows. The byte sequence from controller to the device is: `0x03`, A0, A1, A2, D0, D1, D2, and D3.

TxFIFO Write Sequence to TXD 0	MSB			LSB
First Write	A2	A1	A0	0x03
Second Write	D3	D2	D1	D0

3. **Read Past the Instruction Word.** Read the `qspi.RXFIFO_DATA1` register and receive `0xYYYY_YYYY`.
 - a. `YY = 0`.
4. **Read Flash Memory Data.** Read the `qspi.RXFIFO_DATA2` register and receive `0xD3D2_D1D0`.
 - a. For the second read, software remembers that four bytes are valid.
 - b. Example data: `0x2468ACEF`.
 - c. Overall, software reads these bytes: `0x00`, `0x00`, `0x00`, `0x00`, `0x24`, `0x68`, `0xAC`, `0xEF` and reads the four bytes of data.

The content of the RxFIFO for this example is shown in the table. The byte sequence from the device to controller is: `YY`, `YY`, `YY`, `YY`, `0xEF`, `0xAC`, `0x68` and `0x24`.

RxFIFO Register Reads	MSB			LSB
RXFIFO_DATA1	YY	YY	YY	YY
RXFIFO_DATA2	0x24	0x68	0xAC	0xEF

12.3.6 Register Overview

The register overview is provided in [Table 12-4](#).

Table 12-4: Quad-SPI Register Overview

Address Offset	Mnemonic Software Name	Description
0x00	Config_reg	Configuration
0x04	Intr_status_REG	Interrupt status
0x08	Intrpt_en_REG	Interrupt enable
0x0C	Intrpt_dis_REG	Interrupt disable
0x10	Intrpt_mask_REG	Interrupt mask
0x14	En_REG	SPI enable
0x18	Delay_REG	Delay
0x1C	TXD0	Transmit 4-byte instruction and data
0x20	Rx_data_REG	Receive data (RxFIFO)
0x24	Slave_Idle_count_REG	Slave idle count
0x28	TX_thres_REG	TxFIFO threshold level
0x2C	RX_thres_REG	RxFIFO Threshold level
0x30	GPIO	General purpose inputs and outputs
0x38	LPBK_DLY_ADJ	Loopback master clock delay adjustment
0x80	TXD1	Transmit 1-byte instruction
0x84	TXD2	Transmit 2-byte instruction
0x88	TXD3	Transmit 3-byte instruction
0xA0	LQSPI_CFG	Linear mode configuration
0xA4	LQSPI_STS	Linear mode status
0xFC	MOD_ID	Module ID

12.4 System Functions

12.4.1 Clocks

The controller and I/O interface are driven by the reference clock (QSPI_REF_CLK). The controller's interconnect also requires an APB interface CPU_1x clock. These clocks are generated by the PS clock subsystem.

CPU_1x Clock

Refer to section [25.3 CPU Clock Domains](#), for general clock programming information. The CPU_1x clock runs asynchronous to the CAN reference clock.

Reference Clock

For power management, the clock enable in the slcr register can be used to turn off the clock.

The QSPI_REF_CLK is the main controller clock and is divided down to generate the SCLK clock for the flash memory interface. The QSPI_REF_CLK is sourced from the PS Clock Subsystem. The clock enable, PLL select, and divisor setting are programmed using the slcr.LQSPI_CLK_CTRL register. Refer to section [25.7.3 SDIO, SMC, SPI, Quad-SPI and UART Clocks](#) for Quad-SPI clock programming information. For power management, the clock enable in the slcr register can be used to turn off the clock. The operating frequency for the reference clock is defined in the data sheet.

Clock Ratio Restriction in Manual Mode

In manual mode, the QSPI_REF_CLK frequency must be of greater than or equal value to that of CPU_1x clock frequency for reliable operation of the controller.

There is no such restriction in automatic mode. The reference clock is divided down by qspi.Config_reg[baud_rate_divisor] to generate the SCLK clock for the flash memory.

Example: Setup Reference Clock

This example assumes the selected PLL (ARM, DDR or IO) is operating at 1000 MHz and the desired Quad-SPI reference clock frequency is 200 MHz.

1. **Select PLL source, divisors and enable.** Write 0x0000_0501 to the slcr.QSPI_CLK_CTRL register.
 - a. Enable the reference clock.
 - b. Divide the I/O PLL clock by 5: DIVISOR = 0x05.

Quad-SPI Feedback Clock

The Quad-SPI interface supports an optional feedback clock pin named qspi_sclk_fb_out. This pin is used with the high speed Quad-SPI timing mode. The feedback signal is received from the internal input from the I/O.

When Quad-SPI feedback mode is used, the qspi_sclk_fb_out pin should only be connected to a pull-up or pull-down resistor which is needed to set the MIO voltage mode (vmode).

When operating at a QSPI clock frequency greater than FQSPICLK2, the MIO 8 pin must be programmed as the feedback output clock and the MIO 8 pin must only be connected to a pull-up/pull-down resistor on the PCB for boot strapping.

12.4.2 Resets

The controller has two reset domains: the APB interface and the controller itself. They can be controlled together or independently. The effects for each reset type are summarized in [Table 12-5](#).

Table 12-5: Quad-SPI Reset Effects

Name	APB Interface	TxFIFO and Rx FIFO	Protocol Engine	Registers
ABP Interface Reset slcr.LQSPI_RST_CTRL[LQSPI_CPU1X_RST]	Yes	Yes	No	Yes
PS Reset Subsystem slcr.LQSPI_RST_CTRL[QSPI_REF_RST]	No	Yes	Yes	No

Example: Reset the APB Interface and Quad-SPI Controller

Set and clear controller resets. Write a 1 and then a 0 to the slcr.LQSPI_RST_CTRL[QSPI_REF_RST, LQSPI_CPU1X_RST] bit fields.

12.5 I/O Interface

12.5.1 Wiring Connections

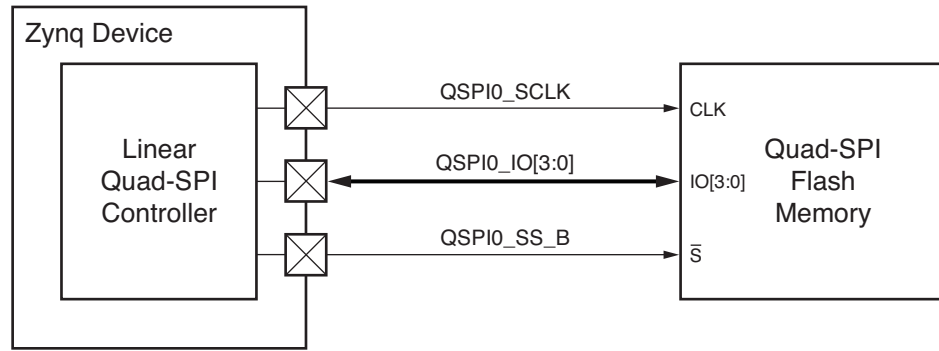
The I/O signals are available via the MIO pins. The Quad-SPI controller supports up to two SPI flash memories in either a shared or separate bus configuration. The controller supports operation in several configurations:

- Quad-SPI single SS 4-bit I/O
- Quad-SPI dual SS, 8-bit parallel I/O
- Quad-SPI dual SS 4-bit stacked I/O
- Quad-SPI single SS, legacy I/O

QSPI 0 should always be present if the QSPI memory subsystem is to be used. QSPI 1 is optional and is only required for the two-memory arrangement. Therefore, QSPI_1 cannot be used alone.

Single SS, 4-bit I/O

A block diagram of the 4-bit flash memory interface connected to the controller configuration is shown in [Figure 12-5](#).

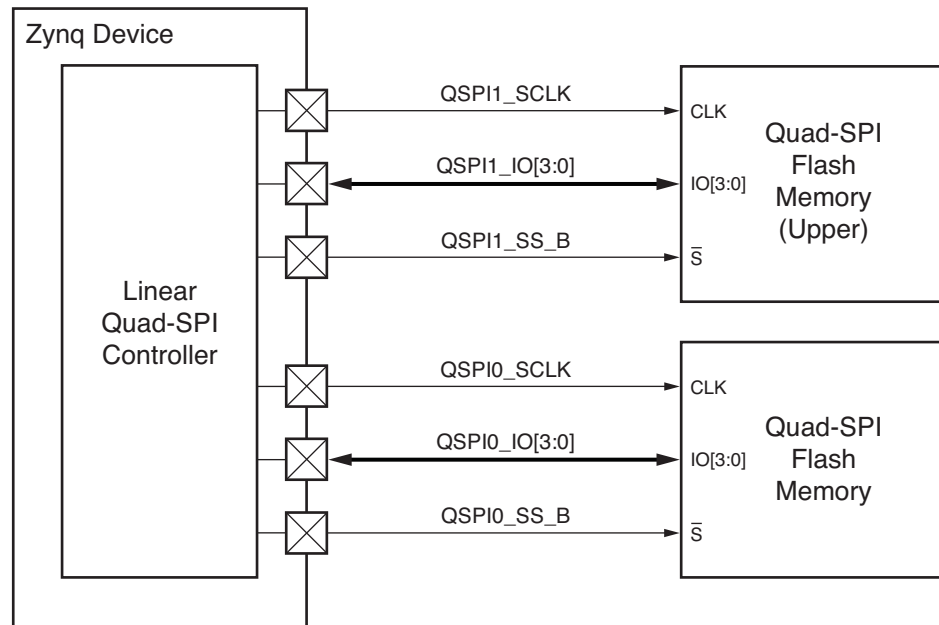


UG585_c12_06_022712

Figure 12-5: Quad-SPI Single SS 4-bit I/O

Dual SS, 4-bit Stacked I/O

The controller supports up to two SPI flash memories operating in parallel, as shown in Figure 12-6. This configuration increases the maximum addressable SPI flash memory from 16 MB (24-bit addressing) to 32 MB (25-bit addressing), and the throughput by a factor of 2.



UG585_c12_07_022712

Figure 12-6: Quad-SPI Dual SS, 8-bit Parallel I/O

Dual SS, 4-bit Stacked I/O

To reduce the I/O pin count, the controller also supports up to two SPI flash memories in a shared bus configuration, as shown in Figure 12-7. This configuration increases the maximum addressable SPI flash memory from 16 MB (24-bit addressing) to 32 MB (25-bit addressing), but the throughput remains the same as for single memory mode. Note that in this configuration, the device level XIP mode (read instruction codes of 0xbb and 0xeb), is not supported.

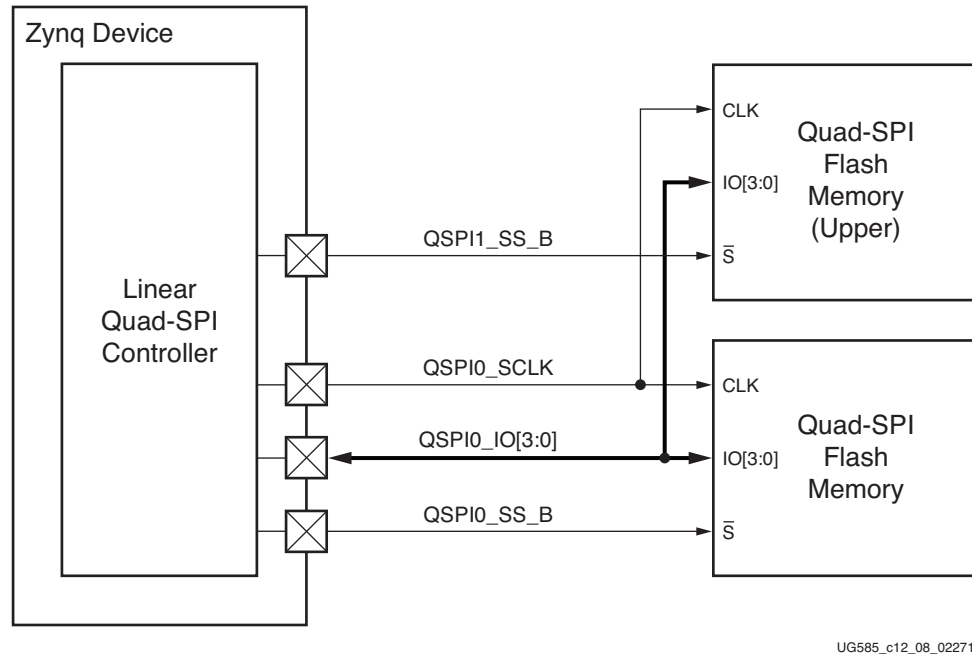


Figure 12-7: Quad-SPI Dual SS 4-bit Stacked I/O

Single SS, Legacy I/O

The Quad-SPI controller can be operated in legacy single-bit serial interface mode for 1x, 2x and 4x I/O modes as shown in Figure 12-8.

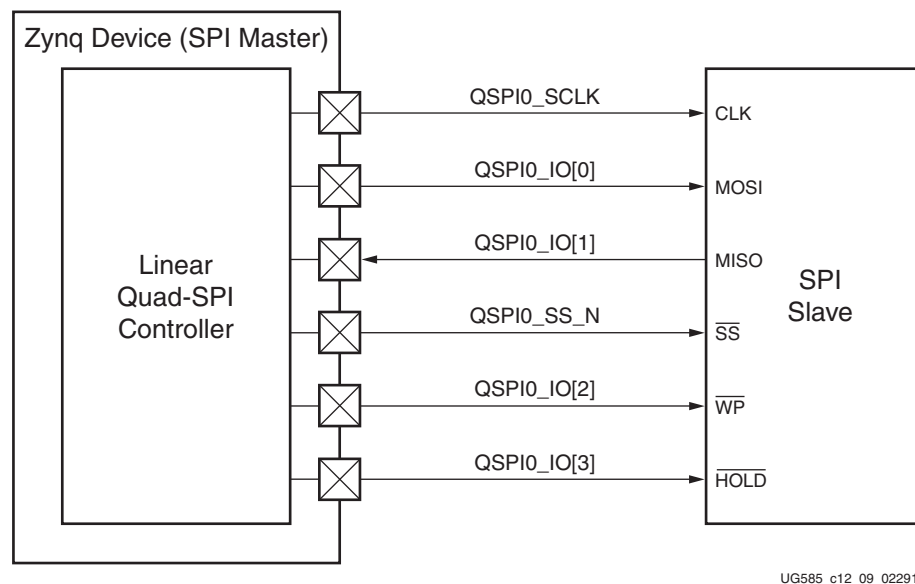


Figure 12-8: Quad-SPI Single SS, Legacy I/O

12.5.2 MIO Programming

The Quad-SPI signals can be routed to specific MIO pins, refer to [Table 12-6](#), Quad-SPI Interface Signals. Wiring diagrams are shown in [Figure 12-5](#) to [Figure 12-8](#). The general routing concepts and MIO I/O buffer configurations are explained in section [2.4 MIO-EMIO](#).

If a four-bit I/O bus is used, then use Quad-SPI 0. If a bus frequency of greater than 40 MHz is needed, then the Quad-SPI feedback clock must be routed on MIN pin 8.

Example: Program I/O for a Single Device

These steps are required for all of the Quad-SPI I/O interface connections listed above.

1. **Configure MIO pin 1 for chip select 0 output.** Write `0x0000_1302` to the `slcr.MIO_PIN_01` register:
 - a. Route Quad-SPI 0 chip select to pin 1.
 - b. 3-state controlled by Quad-SPI (`TRI_ENABLE = 0`).
 - c. LVCMOS18 (refer to the register definition for other voltage options).
 - d. Slow CMOS edge (benign setting).
 - e. Enable internal pull-up resistor.
 - f. Disable HSTL receiver.
2. **Configure MIO pins 2 through 4 for I/O.** Write `0x0000_0302` to each of the `slcr.MIO_PIN_{02:05}` registers:
 - a. Route Quad-SPI 0 I/O pins to pin 2 through 5.
 - b. 3-state controlled by Quad-SPI (`TRI_ENABLE = 0`).
 - c. LVCMOS18 (refer to the register definition for other voltage options).
 - d. Slow CMOS drive edge.
 - e. Disable internal pull-up resistor.
 - f. Disable HSTL receiver.
3. **Configure MIO pin 6 for serial clock 0 output.** Write `0x0000_0302` to the `slcr.MIO_PIN_06` register:
 - a. Route Quad-SPI 0 serial clock to pin 6.
 - b. 3-state controlled by Quad-SPI (`TRI_ENABLE = 0`).
 - c. LVCMOS18 (refer to the register definition for other voltage options).
 - d. Slow CMOS edge (benign setting).
 - e. Disable internal pull-up resistor.
 - f. Disable HSTL receiver.

Option: Add Second Device Chip Select

This step is required for the following I/O connections:

- Dual selects, shared 4-bit data memory interface.

- Dual selects, separate 4-bit data memory interface.
- 4. **Configure MIO pin 0 for chip select 1 output.** Write `0x0000_1302` to the `slcr.MIO_PIN_00` register:
 - a. Route Quad-SPI 1 chip select to pin 0.
 - b. 3-state controlled by Quad-SPI (`TRI_ENABLE = 0`).
 - c. LVCMOS18 (refer to the register definition for other voltage options).
 - d. Slow CMOS edge (benign setting).
 - e. Enable internal pull-up resistor.
 - f. Disable HSTL receiver.

Option: Add Second Serial Clock

This step is required for the Dual Selects, Separate 4-bit Data Memory Interface:

- 5. **Configure MIO pin 9 for serial clock 1 output.** Write `0x0000_0302` to the `slcr.MIO_PIN_09` register:
 - a. Route Quad-SPI 1 serial clock to pin 9.
 - b. 3-state controlled by Quad-SPI (`TRI_ENABLE = 0`).
 - c. LVCMOS18 (refer to the register definition for other voltage options).
 - d. Slow CMOS edge (benign setting).
 - e. Disable internal pull-up resistor.
 - f. Disable HSTL receiver.

Option: Add 4-bit Data

These steps are required for the dual selects, separate 4-bit data memory interface:

- 6. **Configure MIO pins 10 through 13 for I/O.** Write `0x0000_0302` to each of the `slcr.MIO_PIN_{10:13}` registers:
 - a. Route Quad-SPI 1 I/O pins to pin 9 through 13.
 - b. 3-state controlled by Quad-SPI (`TRI_ENABLE = 0`).
 - c. LVCMOS18 (refer to the register definition for other voltage options).
 - d. Slow CMOS drive edge.
 - e. Disable internal pull-up resistor.
 - f. Disable HSTL receiver.

Option: Add Feedback Output Clock

The optional feedback clock is used when the I/O interface is operated above 40 MHz. It should only be connected to a pull-up or pull-down resistor for pin strapping resistor for MIO voltage mode, `vmode`. The feedback clock must also be enabled.

- 7. **Configure MIO pin 8 for feedback clock.** Write `0x0000_0302` to the `slcr.MIO_PIN_08` register:

- a. Route Quad-SPI feedback clock output to pin 8.
- b. 3-state controlled by Quad-SPI (TRI_ENABLE = 0).
- c. LVCMOS18 (refer to the register definition for other voltage options).
- d. Slow CMOS edge (benign setting).
- e. Disable internal pull-up resistor.
- f. Disable HSTL receiver.

12.5.3 MIO Signals

The Quad-SPI flash memory signals are routed through the MIO multiplexer to the MIO device pins. Each side of the dual controller port can be individually enabled or operate together as an 8-bit I/O interface.

The Quad-SPI flash memory signals are routed to the MIO pins as shown in [Table 12-6](#).

Table 12-6: Quad SPI Interface Signals

Quad-SPI Flash Memory Interface				MIO Pin				Controller Default Input Value
Signal	I/O Mode for Data			Quad SPI 0	Quad SPI 1	I/O	Name	
	1-Bit Data	2-Bit Data	4-Bit Data					
Flash Chip Select	~			1	0	O	QSPI{1,0}_SS_B	~
Serial Clock	~			6	9	O	QSPI{1,0}_SCLK	~
Output Feedback Clk	~			8		O	QSPI_SCLK_FB_OUT	~
I/O 0	Master Output	I/O 0	I/O 0	2	10	IO	QSPI{1,0}_IO_0	0
I/O 1	Master Input	I/O 1	I/O 1	3	11	IO	QSPI{1,0}_IO_1	0
I/O 2	Write Protect	Write Protect	I/O 2	4	12	IO	QSPI{1,0}_IO_2	0
I/O 3	Hold	Hold	I/O 3	5	13	IO	QSPI{1,0}_IO_3	0

SD/SDIO Peripheral Controller

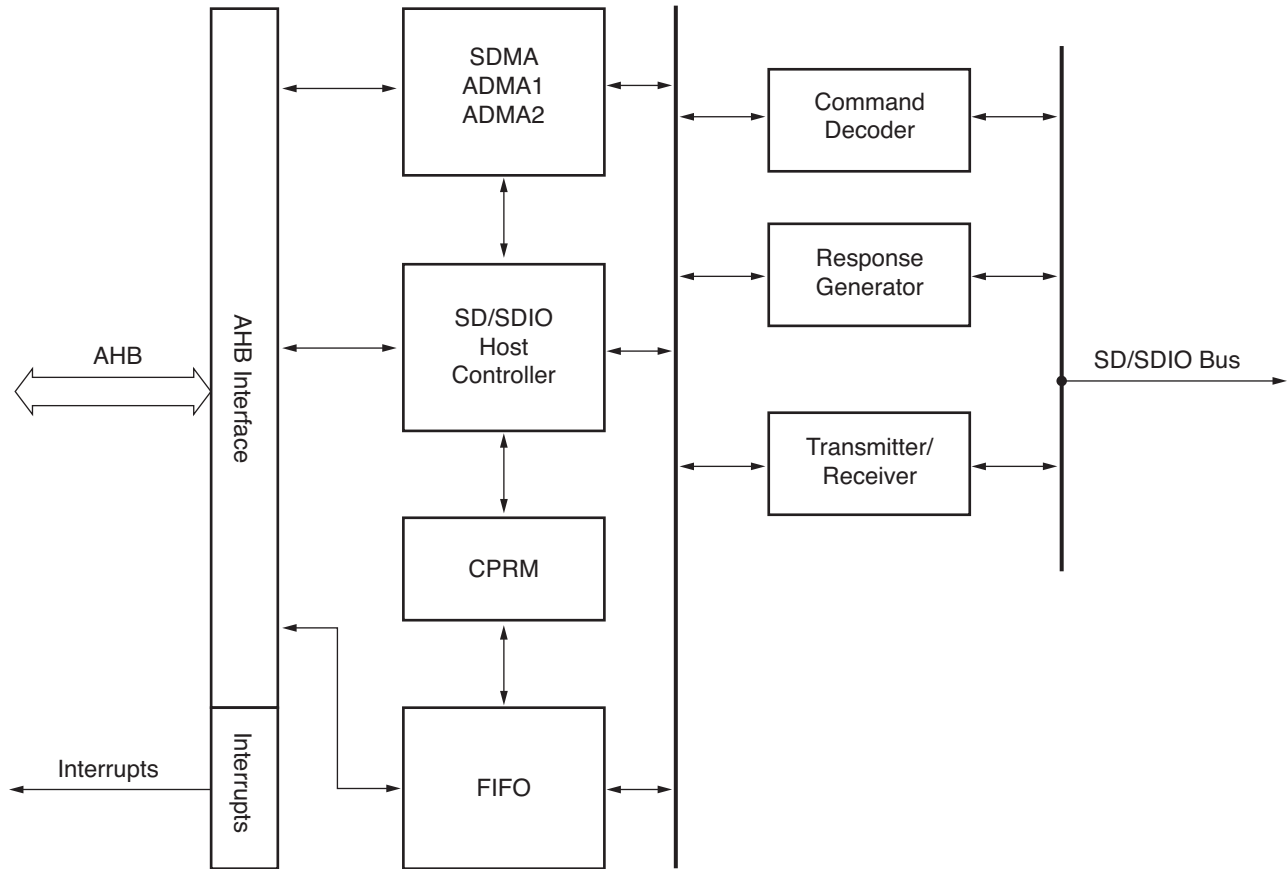
13.1 Introduction

This document expects the user to be familiar with the SD2.0/SDIO 2.0 specifications. Standard specifications are listed in [Appendix A, Additional Resources](#).

The SD/SDIO peripheral controller controls communication with SDIO devices and SD memory cards connected to the PS. It supports SD and SDIO applications in a wide range of portable low-power applications such as 802.11 devices, GPS, WiMAX, UWB, and others. The SD/SDIO peripheral controller block diagram is shown in [Figure 13-1](#).

The SD/SDIO peripheral is compatible with the standard *SD Host Controller Specification Version 2.0 Part A2* with SDMA (single operation DMA), ADMA1 (4 KB boundary limited DMA), and ADMA2 (ADMA2 allows data of any location and any size to be transferred in a 32-bit system memory - scatter-gather DMA) support. The core also supports up to seven functions in SD1, SD4, but does not support SPI mode. It does support SD high-speed (SDHS) and SD High Capacity (SDHC) card standards.

The SD/SDIO peripheral communicates with the ARM processor via the AHB bus. The SDIO device controller supports an internal FIFO to meet throughput requirement.



UG585_c13_01_031812

Figure 13-1: SD/SDIO Peripheral Controller Block Diagram

13.1.1 Key Features

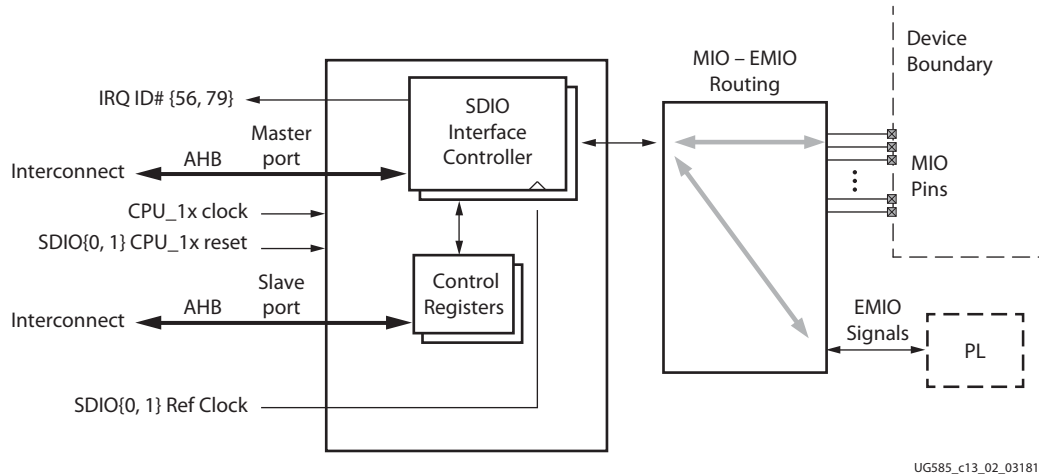
The PS supports the two SD/SDIO devices in the input/output peripherals (IOP) with the following key features:

- Host mode support only
- 100 MHz AHB master-slave CPU 1x clock
- Supports AHB master DMA interface
- Supports AHB slave interface
- Support for version 2.0 of SD specification
- Full speed and low speed support
- 1- and 4-bit data interface support
- Low speed clock 1-400 KHz
- Support for high speed interface
- Full speed clock 1-50 MHz with maximum throughput at 25 MB/s
- Support for memory, I/O, and combination cards

- Support for power control modes
- Support for interrupts
- 1 KB data FIFO interface

13.1.2 System Viewpoint

Figure 13-2 shows the SD/SDIO peripheral controller system viewpoint.



UG585_c13_02_031812

Figure 13-2: SD/SDIO Peripheral Controller System Viewpoint Diagram

13.2 Functional Description

13.2.1 AHB Interface and Interrupt Controller

Data transactions are performed through the AHB interface when the programmed I/O method is used. The processor then uses the Buffer Data Port register for data transfer. The AHB interface initiates read or write transactions with the memory when these transactions are done under DMA control. The controller generates interrupts to the processor if any of the interrupt bits are set in the interrupt status register.

13.2.2 SD/SDIO Host Controller

The SD/SDIO host controller comprises:

- Host-AHB controller
- All control registers
- Bus monitor
- Clock generator

- CRC generator and checker (CRC7 and CRC16)

The host-AHB controller acts as bridge between the AHB bus and the host controller. The SD/SDIO controller registers are programmed by the processor through the AHB interface. Interrupts are generated based on the values set in the Interrupt Status and Interrupt Enable registers.

The bus monitor checks for violations occurring on the SD bus and timeout conditions. The clock generation block generates the SD clock depending on the value programmed in the Clock Control register.

The CRC7 and CRC16 generators calculate the CRC for command and data transfers to the SD/SDIO card. The CRC7 and CRC16 checker checks for any CRC errors in the response and data received from the SD/SDIO card. To detect data defects on the card, the host can include error correction codes in the payload data. ECC code is used to store data on the card. This ECC code is used by the application to decode the user data.

13.2.3 Data FIFO

The controller uses two 512 byte dual port FIFOs for performing both write and read transactions. During a write transaction (data is transferred from the processor to an attached card) data is written by the processor alternatively into the first and second FIFO. When data is transferred to an attached card, alternatively the second and first FIFO is used, providing maximum data throughput.

During a read transaction (data is transferred from an attached card to the processor) the data from the card is alternatively written in the two FIFOs. When data from one FIFO is transferred to the processor the second FIFO is filled with data from the card and vice versa, optimizing data throughput.

If the controller cannot accept any data from a connected card, it issues a "read wait", stopping the data transfer from the card by stopping the clock.

13.2.4 Command and Control Logic

The control logic block transmits data on the data lines during a write transaction and receives data during read transaction. The command control logic block transmits commands on the command lines and receives the response from the SD2.0 or SDIO2.0.

13.2.5 Bus Monitor

The bus monitor checks for violations occurring in the SD bus, and timeout conditions.

13.2.6 Stream Write and Read

This functionality applies to both DMA and non-DMA modes.

WRITE_DAT_UNTIL_STOP(CMD20) writes a data stream from the host, starting at the given address, until a STOP_TRANSMISSION follows.

READ_DAT_UNTIL_STOP(CMD11) reads a data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.

The host controller switches to the second FIFO after writing/reading a block of data to/from the first FIFO, but the stream transaction block size is not be programmed by the driver. So for both stream write and stream read transactions, it is recommended that the host driver writes the maximum FIFO size value to the Block Size register. Because the SDIO FIFO slice is set to 512 bytes, the host driver must write 512 bytes to the Block Size register. Therefore FIFO switching occurs after writing/reading the 512 bytes of data.

13.2.7 Clocks

The SDIO clock is derived from SDIO reference clock based on the Clock Control register value programmed by the driver and is available only when the SD clock enable is set by the driver. The maximum frequency is 50 MHz for SD.

The host controller supports both full speed and high speed cards. For the high speed card, the host controller should clock out the data at the rising edge of the SDIO clock. For the full speed card, the host controller should clock out the data at the falling edge of the SDIO clock.

13.2.8 Soft Resets

The host controller supports all soft resets mentioned in the *SD2.0/SDIO2.0 Host Controller Specification*.

13.2.9 FIFO Overrun and Underrun Conditions

This functionality applies to both DMA and non-DMA modes.

Write

During the write transaction, the host controller transmits data to the card only when a block of data is ready to transmit and the card is not busy. Therefore an under-run condition cannot occur in the SD side.

- In DMA mode, the host controller initiates a DMA READ from the ARM processor only if space is available to accept a block of data.
- In non-DMA mode, the host controller asserts a buffer write ready interrupt only if space is available to accept a block of data.

Read

During the read transaction when the FIFO is full (the FIFO does not have enough space to accept a block of data from the card) the host controller stops the `clk_sd` to the card. Therefore an over-run condition cannot occur in SD side.

- In DMA mode the host controller initiates a DMA WRITE to the ARM processor only on reception of a block of data from card.
- In non-DMA mode, the host controller asserts a buffer read ready interrupt only on reception of a block of data from card.

13.3 Protocols

13.3.1 Data Transfer Protocol Overview

SD transfers are basically classified into following three types according to how the number of blocks is specified:

Single Block Transfer

The number of blocks is specified to the host controller before the transfer. The number of blocks specified is always one.

Multiple Block Transfer

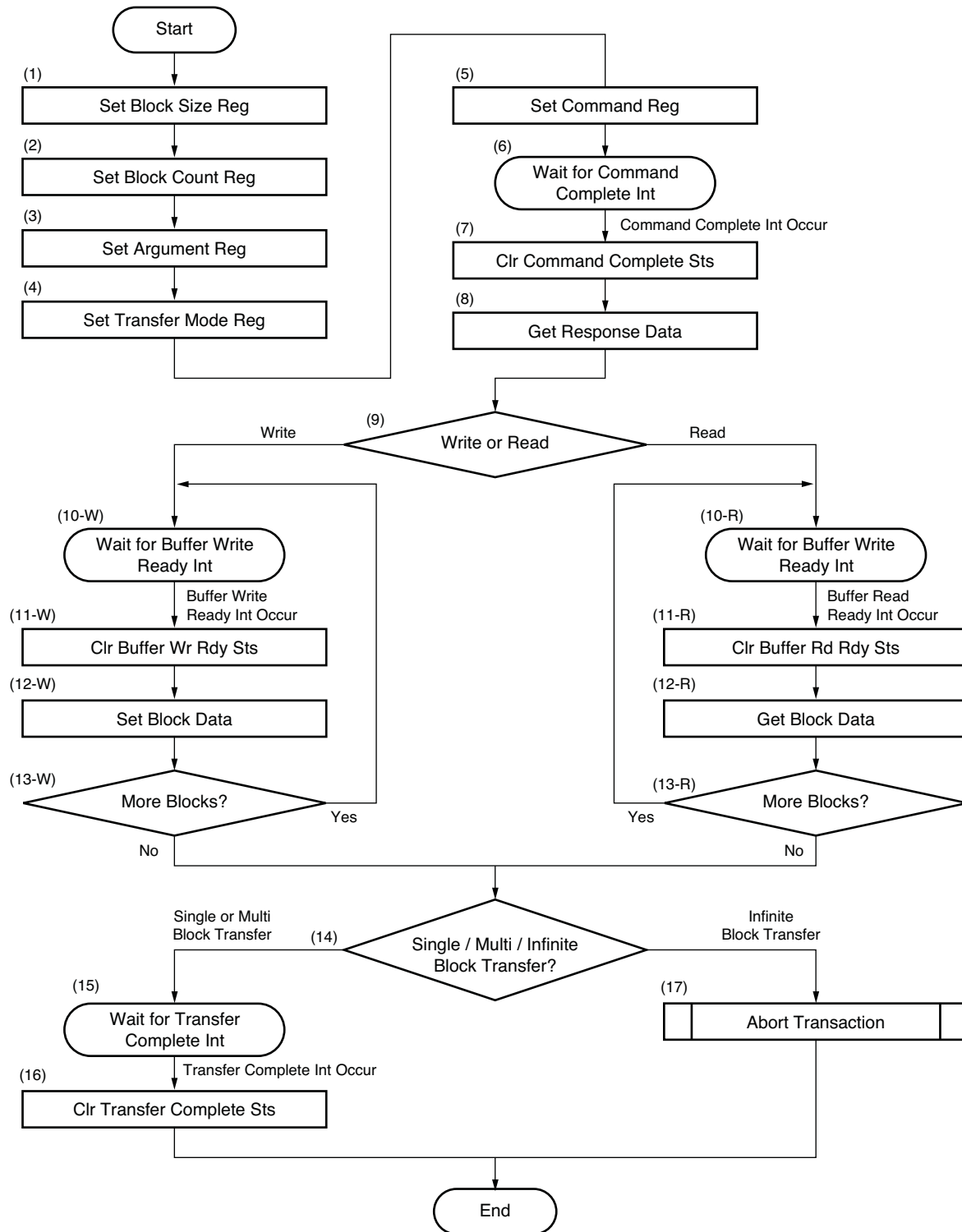
The number of blocks is specified to the host controller before the transfer. The number of blocks specified is one or more.

Infinite Block Transfer

The number of blocks is not specified to the host controller before the transfer. This transfer is continued until an abort transaction is executed. This abort transaction is performed by CMD12 in the case of an SD memory card and by CMD52 in the case of an SDIO card.

13.3.2 Data Transfers Without DMA

Figure 13-3 shows data transfers without using DMA.



UG585_c13_03_031812

Figure 13-3: Data Transfer Using DAT Line Sequence (Without Using DMA)

The sequence for data transfers without using DMA is as follows:

1. Set the value corresponding to the executed data byte length of one block to the Block Size register.
 2. Set the value corresponding to the executed data block count to the Block Count register.
 3. Set the value corresponding to the issued command to the Argument register.
 4. Set the value to Multi/Single Block Select and Block Count Enable. Set the value corresponding to the issued command to Data Transfer Direction, Auto CMD12 Enable, and DMA Enable.
 5. Set the value corresponding to the issued command to the Command register.
- Note:** When writing the upper byte of the command register, the SD command is issued.
6. Wait for the command complete interrupt.
 7. Write a 1 to the Command Complete bit in the Normal Interrupt Status register to clear this bit.
 8. Read the Response register and get the necessary information in accordance with the issued command.
 9. In the case where this sequence is for writing to a card, go to Step (10-W). In case of read from a card, go to Step (10-R).

(10-W). Wait for a buffer write ready interrupt.

- Non-DMA Write Transfer

On receiving the buffer write ready interrupt the ARM processor acts as a master and starts transferring the data via the Buffer Data Port register (FIFO_1). The transmitter starts sending the data on the SD bus when a block of data is ready in FIFO_1. While transmitting the data on the SD bus the buffer write ready interrupt is sent to the ARM processor for the second block of data. The ARM processor acts as a master and starts sending the second block of data via the buffer data port register to FIFO_2. The buffer write ready interrupt is asserted only when a FIFO is empty and available to receive a block of data.

(11-W). Write a 1 to the Buffer Write Ready bit in the Normal Interrupt Status register to clear this bit.

(12-W). Write a block of data (according to the number of bytes specified in Step (1)) to the Buffer Data Port register.

(13-W). Repeat until all blocks are sent and then go to Step (14).

- Non-DMA Read Transfer

A buffer read ready interrupt is asserted whenever a block of data is ready in one of the FIFOs. On receiving the buffer read ready interrupt, the ARM processor acts as a master and starts reading the data via the Buffer Data Port register (FIFO_1). The receiver starts reading the data from the SD bus only when a FIFO is empty and available to receive a block of data. When both of the FIFOs are full the host controller stops the data coming from the card by means of a read wait mechanism (if the card supports read wait) or through clock stopping.

(10-R). Wait for a buffer read ready interrupt

(11-R). Write a 1 to the Buffer Read Ready bit in the Normal Interrupt Status register to clear this bit.

- (12-R). Read a block of data (according to the number of bytes specified in Step (1)) from the Buffer Data Port register.
- (13-R). Repeat until all blocks are received and then go to Step (14).
- 14. If this sequence is for a single or multiple block transfer, go to Step (15). In case of an infinite block transfer, go to Step (17).
- 10. Wait for a transfer complete interrupt.
- 11. Write a 1 to the Transfer Complete bit in the Normal Interrupt Status register to clear this bit.
- 12. Perform the sequence for abort transaction.

Note: Step (1) and Step (2) can be executed at same time. Step (4) and Step (5) can be executed at same time.

13.3.3 Using DMA

Figure 13-4 shows data transfers using DMA.

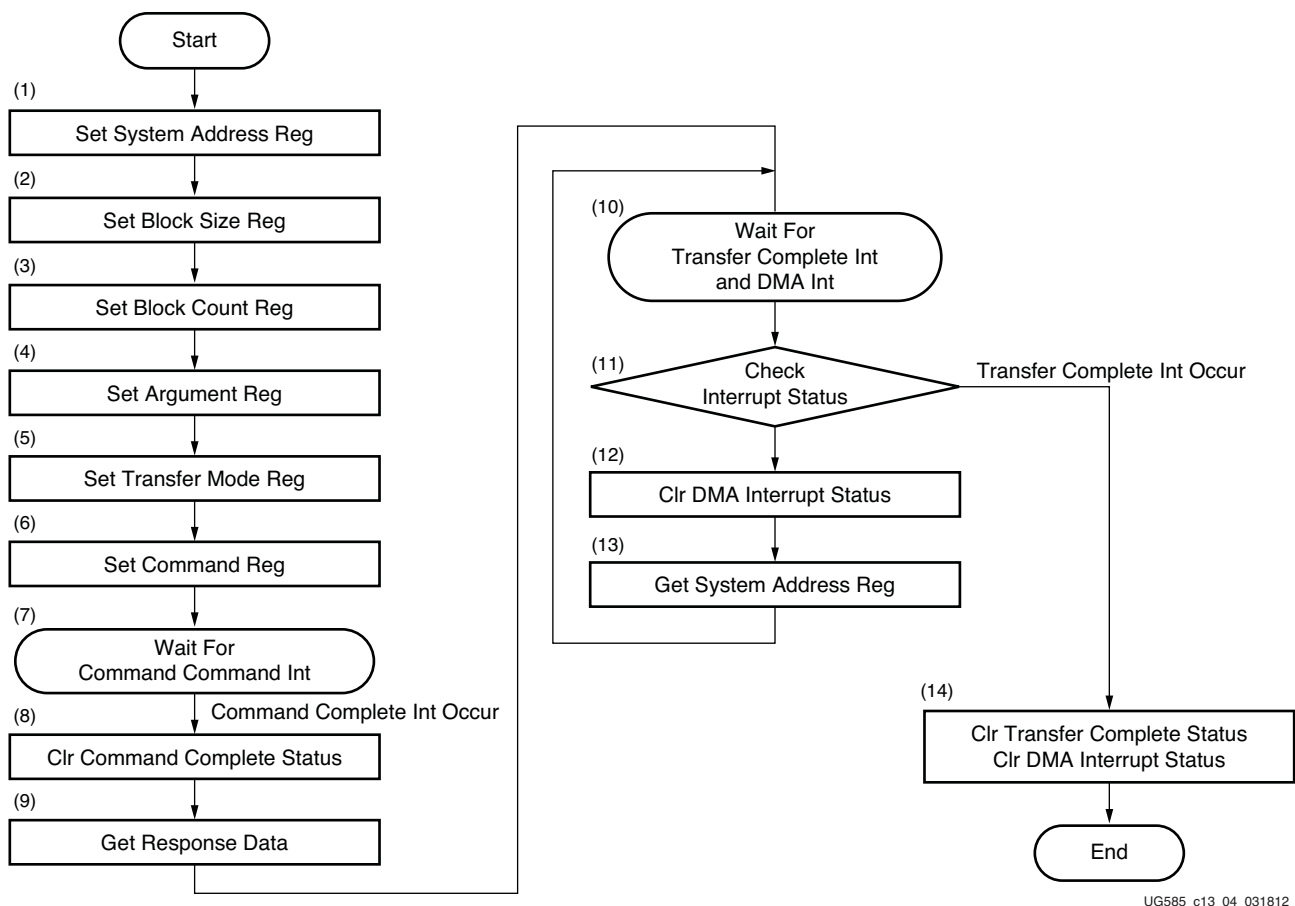


Figure 13-4: Data Transfer Using DMA

Burst types such as an 8-beat incrementing burst or a 4-beat incrementing burst, or a single transfer is used to transfer or receive the data from the system memory mainly to avoid the hold of the AHB bus by the master for a longer time.

The sequence for using DMA is as follows:

1. Set the system address for DMA in the System Address register.
 2. Set the value corresponding to the executed data byte length of one block in the Block Size register.
 3. Set the value corresponding to the executed data block count in the Block Count register.
 4. Set the value corresponding to the issued command to the Argument register.
 5. Set the value to Multi/Single Block Select and Block Count Enable. Set the value corresponding to the issued command to Data Transfer Direction, Auto CMD12 Enable, and DMA Enable.
 6. Set the value corresponding to the issued command to the Command register.
- Note:** When writing to the upper byte of the Command register, the SD command is issued.
7. Wait for the command complete interrupt.
 8. Write a 1 to the Command Complete in the Normal Interrupt Status register for clearing this bit.
 9. Read the Response register and get the necessary information in accordance with the issued command.

- DMA Read Transfer

On receiving the Response End Bit from the card for the write command (data is flowing from the host to the card) the SD host controller acts as the master and requests the AHB bus. After receiving the grant the host controller starts reading a block of data from system memory and fills the first half of the FIFO. Whenever a block of data is ready the transmitter starts sending the data on the SD bus.

While transmitting the data on the SD bus the host controller requests the bus to fill the second block in the second half of the FIFO. "Ping Pong" FIFOs are used to increase the throughput. Similarly, the host controller reads a block of data from system memory whenever a FIFO is empty. This continues until all of the blocks are read from system memory. A transfer complete interrupt is set only after transferring all of the blocks of data to the card.

- DMA Write Transfer

The block of data received from the card (data is flowing from the card to the host) is stored in the first half of the FIFO. Whenever a block of data is ready the SD host controller acts as the master and request the AHB bus. After receiving the grant the host controller starts writing a block of data into system memory from the first half of the FIFO.

While transmitting data into system memory the host controller receives the second block of data and stores it in the second half of the FIFO. Similarly the host controller writes a block of data into system memory whenever data is ready. This continues until all of the blocks are transferred to system memory. A transfer complete interrupt is set only after transferring all of the blocks of data into system memory.

Note: The host controller receives a block of data from the card only when it has room to store a block of data in the FIFO. When both FIFOs are full the host controller stops the data coming from the card through a "read wait" mechanism (if the card supports read wait) or through clock stopping.

10. Wait for the transfer complete interrupt and DMA interrupt.
11. If Transfer Complete is set to 1, go to Step (14). If DMA Interrupt is set to 1 go to Step (12). Transfer Complete has a higher priority than DMA Interrupt.
12. Write a 1 to the DMA Interrupt bit in the Normal Interrupt Status register to clear this bit.
13. Set the next system address of the next data position to the System Address register and go to Step (10).
14. Write a 1 to the Transfer Complete and DMA Interrupt in the Normal Interrupt Status register to clear this bit.

Note: Step (2) and Step (3) can be executed at same time. Step (5) and Step (6) can also be executed at same time.

For example, if the host wants to transfer 4 KB of data to the card and assuming the maximum block size is 256 bytes, the host driver programs the Block Size register as 256 and Block Count register with the value 16. The AHB master and transmitter residing inside the SD2.0/SDIO2.0 host controller get the information (how much data to transfer) from these registers. Using the above information, the AHB master acts as a master and initiates a data read transaction (to read a block of data — 256 bytes from the system memory).

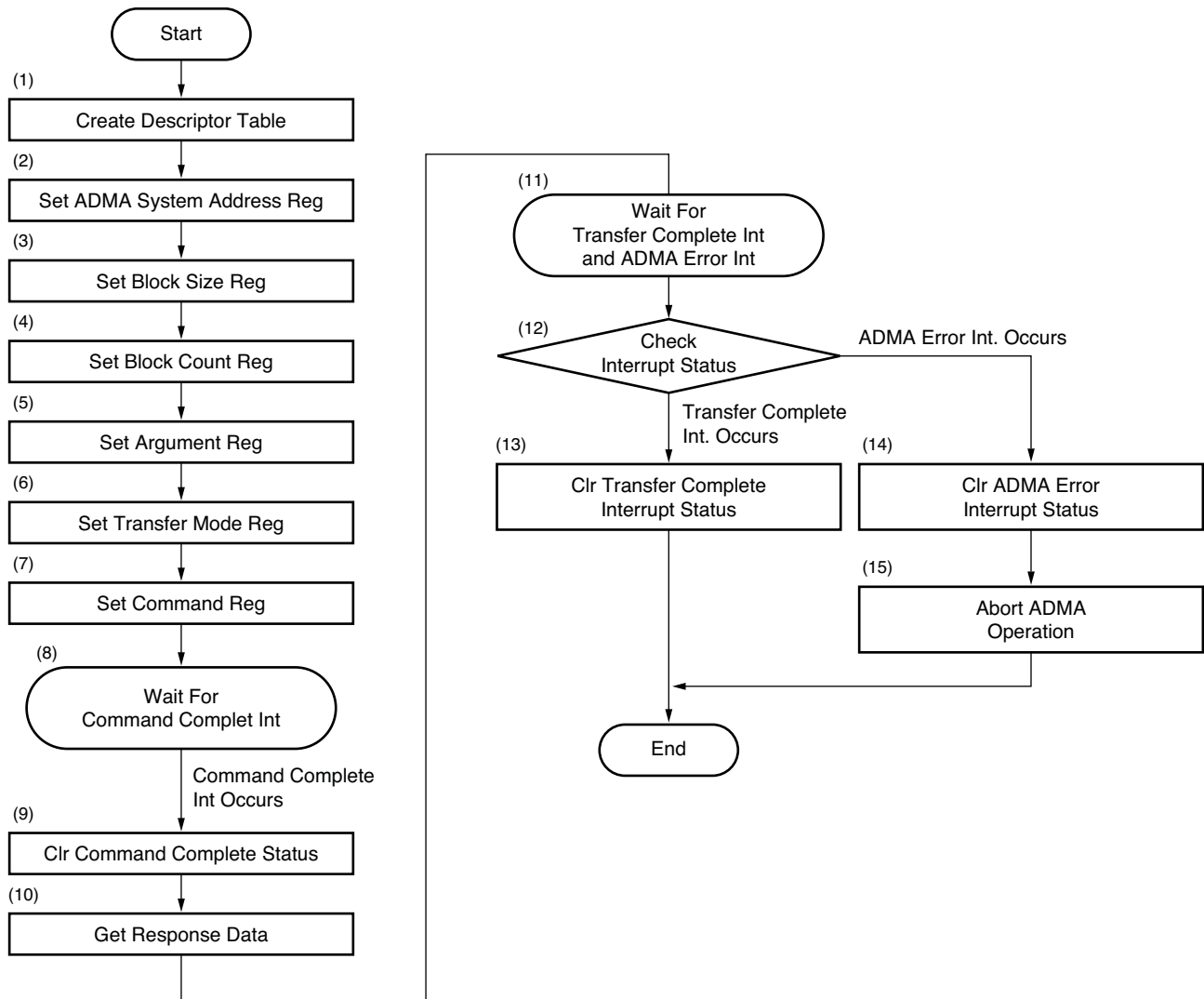
The following types of burst are used mainly to avoid hold of the AHB bus by the master for a longer time.

- Single transfer
- 4-beat incrementing burst
- 8-beat incrementing burst

The first block of data is received in the first half of the FIFO and the second block in the second half of the FIFO. Similarly, the remaining blocks are received in alternate FIFOs. Whenever a block of data is ready in FIFO, the transmitter starts transmitting the block of data (256 bytes) onto the SD bus. After transmitting the entire block of data to the card, the transmitter waits for a status response from the card. Transmitter sends the next block of data only when it receives a good status response from the card for the previous block of data, otherwise the transaction is aborted and the host goes for a fresh transaction.

13.3.4 Using ADMA

Figure 13-5 shows data transfers using ADMA.



UG585_c13_05_031812

Figure 13-5: Data Transfer Using ADMA

The sequence for using ADMA is as follows

1. Create a descriptor table for ADMA in system memory.
2. Set the descriptor address for ADMA in the ADMA System Address register.
3. Set the value corresponding to the executed data byte length of one block in the Block Size register.
4. Set the value corresponding to the executed data block count in the Block Count register in accordance with SDIO register map.

If the Block Count Enable bit in the Transfer Mode register is set to 1, the total data length can be designated by the Block Count register and the descriptor table. These two parameters shall indicate same data length. However, transfer length is limited by the 16-bit Block Count register. If the Block Count Enable bit in the Transfer Mode register is set to 0, the total data length is designated not by Block Count register, but the descriptor table. In this case, if the ADMA reads more data than the length programmed in the descriptor from the SD card, the operation is aborted asynchronously and the extra read data is discarded when the ADMA is completed.

5. Set the argument value to the Argument register.
6. Set the value to the Transfer Mode register. The host driver determines Multi/Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable and DMA Enable. Multi/Single Block Select and Block Count Enable are determined according to SDIO register map.
7. Set the value to the Command register.
Note: When writing to the upper byte [3] of the Command register, the SD command is issued and DMA is started.
8. Wait for the command complete interrupt.
9. Write a 1 to the Command Complete bit in the Normal Interrupt Status register to clear this bit.
10. Read the Response register and get the necessary information from the issued command.
11. Wait for the transfer complete interrupt and ADMA error interrupt.
12. If the Transfer Complete is set to 1, go to Step (13). If the ADMA Error Interrupt is set to 1, go to Step (14).
13. Write a 1 to the Transfer Complete Status bit in the Normal Interrupt Status register to clear this bit.
14. Write a 1 to the ADMA Error Interrupt Status bit in the Error Interrupt Status register to clear this bit.
15. Abort ADMA operation. SD card operation should be stopped by issuing an abort command. If necessary, the host driver checks the ADMA Error Status register to detect why the ADMA error is generated.

Note: Step (3) and Step (4) can be executed simultaneously. Step (6) and Step (7) can also be executed simultaneously.

Note: During ADMA2 operation, the controller will not generate a DMA interrupt if the INT attribute is set along with NOP, RSVD, or LINK attribute.

13.3.5 Abort Transaction

An abort transaction is performed using CMD12 for a SD memory card and by using CMD52 for a SDIO card. There are two cases where the HD needs to do an abort transaction:

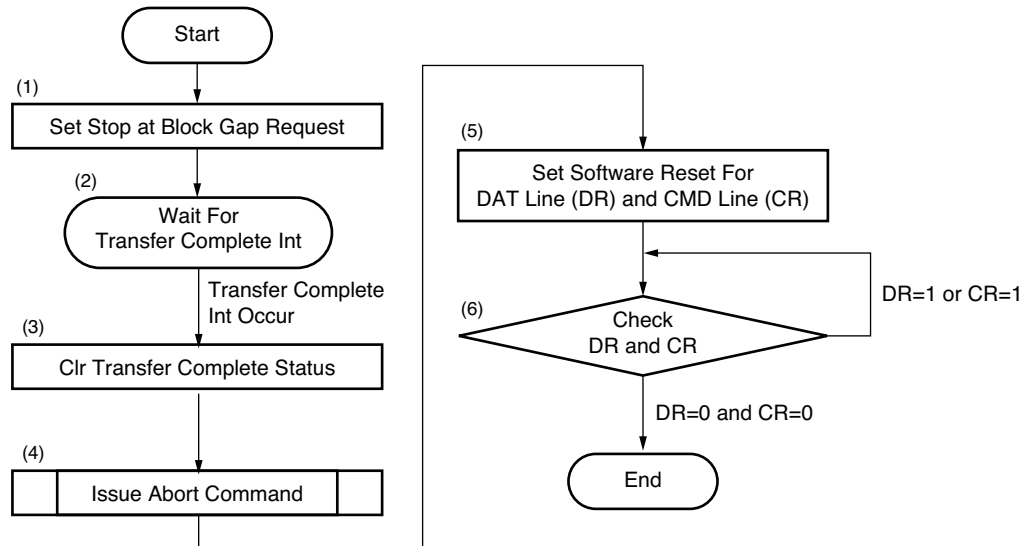
- When the HD stops infinite block transfers.
- When the HD stops transfers while a multiple block transfer is executing.

There are two ways to issue an abort command. The first is an asynchronous abort. The second is a synchronous abort. In an asynchronous abort sequence, the HD can issue an abort command at anytime unless the Command Inhibit (CMD) bit in the Present State register is set to 1. In a synchronous abort, the HD issues an abort command after the data transfer stopped via the Stop At Block Gap Request bit in the Block Gap Control register.

Synchronous Abort

The following sequence performs a synchronous abort.

1. Set the Stop At Block Gap Request bit in the Block Gap Control register to 1 to stop SD transactions.
2. Wait for a transfer complete interrupt.



UG585_c13_06_031812

Figure 13-6: Synchronous Abort Sequence

3. Set the Transfer Complete bit to 1 in the Normal Interrupt Status register to clear this bit.
4. Issue an abort command.
5. Set both the Software Reset for DAT Line and Software Reset for CMD Line bits to 1 in the Software Reset register to do a software reset.
6. Check the Software Reset for DAT Line and Software Reset for CMD Line in the Software Reset register. If both Software Reset for DAT Line and Software Reset for CMD Line are 0, go to "END". If either the Software Reset for DAT Line or the Software Reset for CMD Line is 1, repeat Step (6).

13.3.6 External Interface Usage Example

Zynq-7000 devices provide two secure digital (SD) ports that support SD and SDIO devices (see [Figure 13-7](#)).

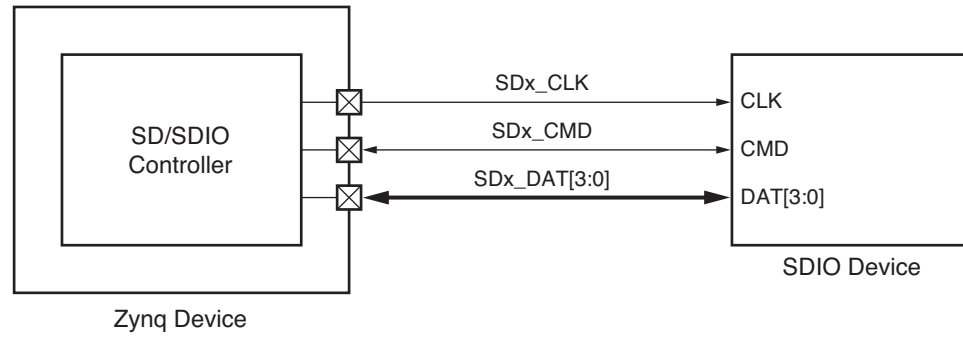


Figure 13-7: SDIO Block Diagram

13.3.7 Supported Configurations

The SD/SDIO controller supports operation in several configurations:

- Secure digital (SD) memory
- Secure digital input/output (SDIO)

Some SD card slots provide two additional pins: card detect (CD) to signal the insertion or presence of a card and write protect (WP) to report the position of the write protect switch on memory cards. These pins are usually pulled to GND when a card is detected or the card is write-protected. They need to be pulled up to the MIO I/O voltage with a 50 K Ω resistor (see Figure 13-8).

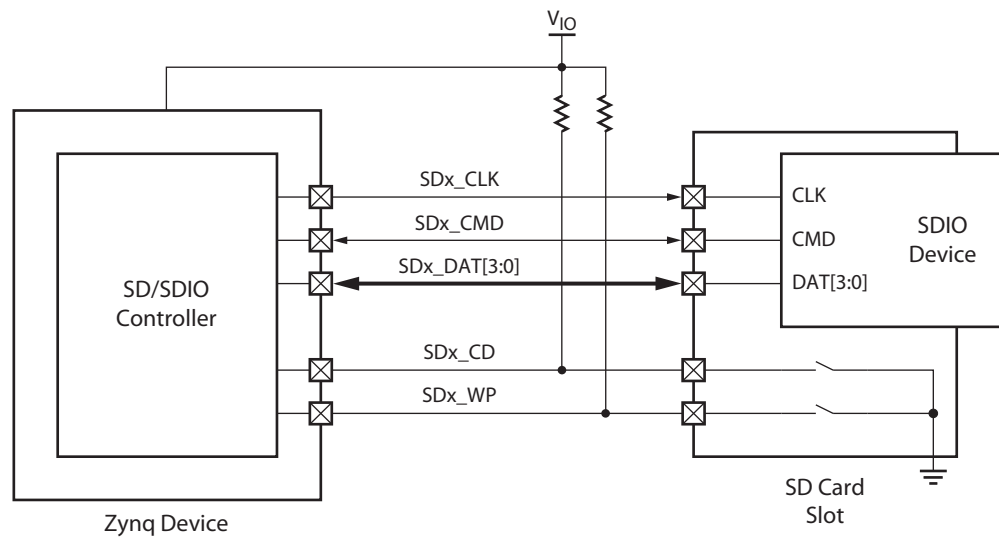


Figure 13-8: SD/SDIO Card Detect and Write Protect

13.3.8 Bus Voltage Translation

The SDIO power pin SDx_POW can be used to control power to the SDIO slots. Depending on the I/O voltage for the selected MIO bank and the SD/SDIO devices connected to the bus, it might be necessary to use a level translator.

13.4 SDIO Controller Media Interface Signals

The SDIO media interface signals are independently routed to the MIO pins or to a set of EMIO interface signals, see [Table 13-1](#). The MIO pins and any restrictions based on device version are shown in the MIO table in section [2.4.4 MIO-at-a-Glance Table](#).

Table 13-1: SDIO Interface Signals

SDIO Interface	Default Controller Input Value	MIO Pin		EMIO Signal	
		Number	I/O	Name	I/O
SDIO 0 Clock	0	16, 28, 40	IO	EMIOSDIO0CLKFB	I
	~			EMIOSDIO0CLK	O
SDIO 0 Command	0	17, 29, 41	IO	EMIOSDIO0CMDI	I
	~			EMIOSDIO0CMDO	O
	~			EMIOSDIO0CMDTN	O
SDIO 0 Data 0	0	18, 30, 42	IO	EMIOSDIO0DATAI0	I
	~			EMIOSDIO0DATAO0	O
	~			EMIOSDIO0DATATN0	O
SDIO 0 Data 1	0	19, 31, 43	IO	EMIOSDIO0DATAI1	I
	~			EMIOSDIO0DATAO1	O
	~			EMIOSDIO0DATATN1	O
SDIO 0 Data 2	0	20, 32, 44	IO	EMIOSDIO0DATAI2	I
	~			EMIOSDIO0DATAO2	O
	~			EMIOSDIO0DATATN2	O
SDIO 0 Data 3	0	21, 33, 45	IO	EMIOSDIO0DATAI3	I
	~			EMIOSDIO0DATAO3	O
	~			EMIOSDIO0DATATN3	O
SDIO 0 Card Detect		Any pin except 7 and 8	I	EMIOSDIO0CDN	I
SDIO 0 Write Protect		Any pin except 7 and 8	I	EMIOSDIO0WVP	I
SDIO 0 Power Control	~	Any even pin	O	EMIOSDIO0BUSPOW	O
SDIO 0 LED Control	~	~	~	EMIOSDIO0LED	O
SDIO 0 Bus Voltage	~	~	~	EMIOSDIO0BUSVOLT[2:0]	O

Table 13-1: SDIO Interface Signals (Cont'd)

SDIO Interface	Default Controller Input Value	MIO Pin		EMIO Signal	
		Number	I/O	Name	I/O
SDIO 1 Clock	0	12, 24, 36, 48	IO	EMIOSDIO1CLKFB	I
	~			EMIOSDIO1CLK	O
SDIO 1 Command	0	11, 23, 35, 47	IO	EMIOSDIO1CMDI	I
	~			EMIOSDIO1CMDO	O
	~			EMIOSDIO1CMDTN	O
SDIO 1 Data 0	0	10, 22, 34, 46	IO	EMIOSDIO1DATAI0	I
	~			EMIOSDIO1DATAO0	O
	~			EMIOSDIO1DATATN0	O
SDIO 1 Data 1	0	13, 25, 37, 49	IO	EMIOSDIO1DATAI1	I
	~			EMIOSDIO1DATAO1	O
	~			EMIOSDIO1DATATN1	O
SDIO 1 Data 2	0	14, 26, 38, 50	IO	EMIOSDIO1DATAI2	I
	~			EMIOSDIO1DATAO2	O
	~			EMIOSDIO1DATATN2	O
SDIO 1 Data 3	0	15, 27, 39, 51	IO	EMIOSDIO1DATAI3	I
	~			EMIOSDIO1DATAO3	O
	~			EMIOSDIO1DATATN3	O
SDIO 1 Card Detect		Any pin except 7 and 8	I	EMIOSDIO1CDN	I
SDIO 1 Write Protect		Any pin except 7 and 8	I	EMIOSDIO1WP	I
SDIO 1 Power Control	~	Any odd pin	O	EMIOSDIO1BUSPOW	O
SDIO 1 LED Control	~	~	~	EMIOSDIO1LED	O
SDIO 1 Bus Voltage	~	~	~	EMIOSDIO1BUSVOLT[2:0]	O

13.4.1 SDIO EMIO Considerations

The SDIO interfaces are enabled through the MIO as well as the EMIO interface. They are mutually exclusive in that once you route the interface through the MIO it no longer is available via the EMIO. If the designer chooses to use the EMIO interface for SDIO due to other MIO priorities, the designer should know that the maximum operating frequency for SDIO via EMIO is limited. The EMIO connectivity should connect the PS SDIO EMIO interface directly to PL PADs. Additionally, it might be necessary to restrict the SDIO to operate in full or low speed modes.

General Purpose I/O (GPIO)

14.1 Introduction

The general purpose I/O (GPIO) peripheral provides software with observation and control of up to 54 device pins via the MIO module. It also provides access to 64 inputs from the Programmable Logic (PL) and 128 outputs to the PL through the EMIO interface. The GPIO is organized into four banks of registers that group related interface signals.

Each GPIO is independently and dynamically programmed as input, output, or interrupt sensing. Software can read all GPIO values within a bank using a single load instruction, or write data to one or more GPIOs (within a range of GPIOs) using a single store instruction. The GPIO control and status registers are memory mapped at base address `0xE000_A000`.

Key features of the GPIO peripheral are summarized as follows:

- 54 GPIO signals for device pins (routed through the MIO multiplexer)
 - Outputs are 3-state capable
- 192 GPIO signals between the PS and PL via the EMIO interface
 - 64 Inputs, 128 outputs (64 true outputs and 64 output enables)
- The function of each GPIO can be dynamically programmed on an individual or group basis
- Enable, bit or bank data write, output enable and direction controls
- Programmable interrupts on individual GPIO basis
 - Status read of raw and masked interrupt
 - Selectable sensitivity: Level-sensitive (High or Low) or edge-sensitive (positive, negative, or both)

14.2 Block Diagram

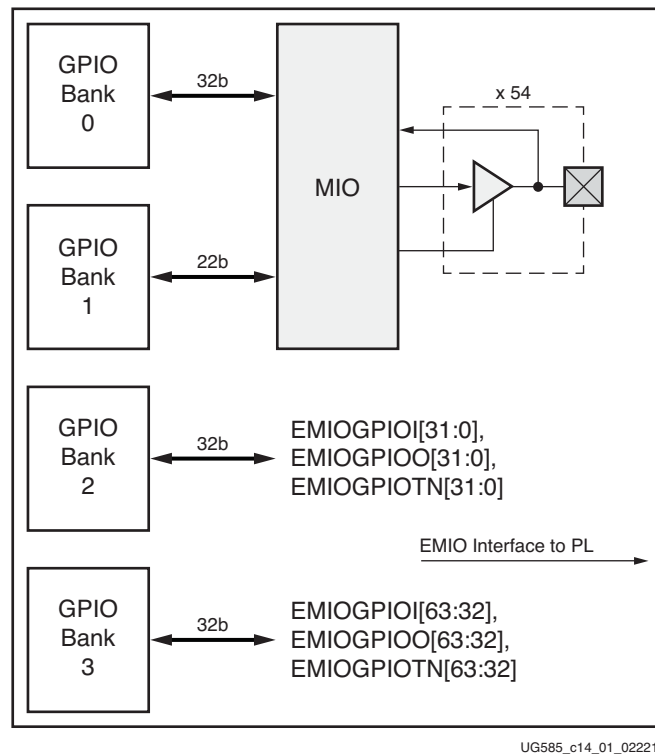


Figure 14-1: GPIO Block Diagram

As shown in Figure 14-1, the GPIO module is divided into four banks:

- Bank0: 32-bit bank controlling MIO pins[31:0]
- Bank1: 22-bit bank controlling MIO pins[53:32]
Note: Bank1 is limited to 22 bits because the MIO has a total of 54 pins.
- Bank2: 32-bit bank controlling EMIO signals[31:0]
- Bank3: 32-bit bank controlling EMIO signals[63:32]

The GPIO is controlled by software through a series of memory-mapped registers. The control for each bank is the same, although there are minor differences between the MIO and EMIO banks due to their differing functionality.

Restrictions

The 7z010 CLG225 device supports 32 MIO pins as shown in the MIO table in section [2.4.4 MIO-at-a-Glance Table](#).

14.3 GPIO Control of Device Pins

This section describes the operation of Bank0 and Bank1 (see [Figure 14-2](#)).

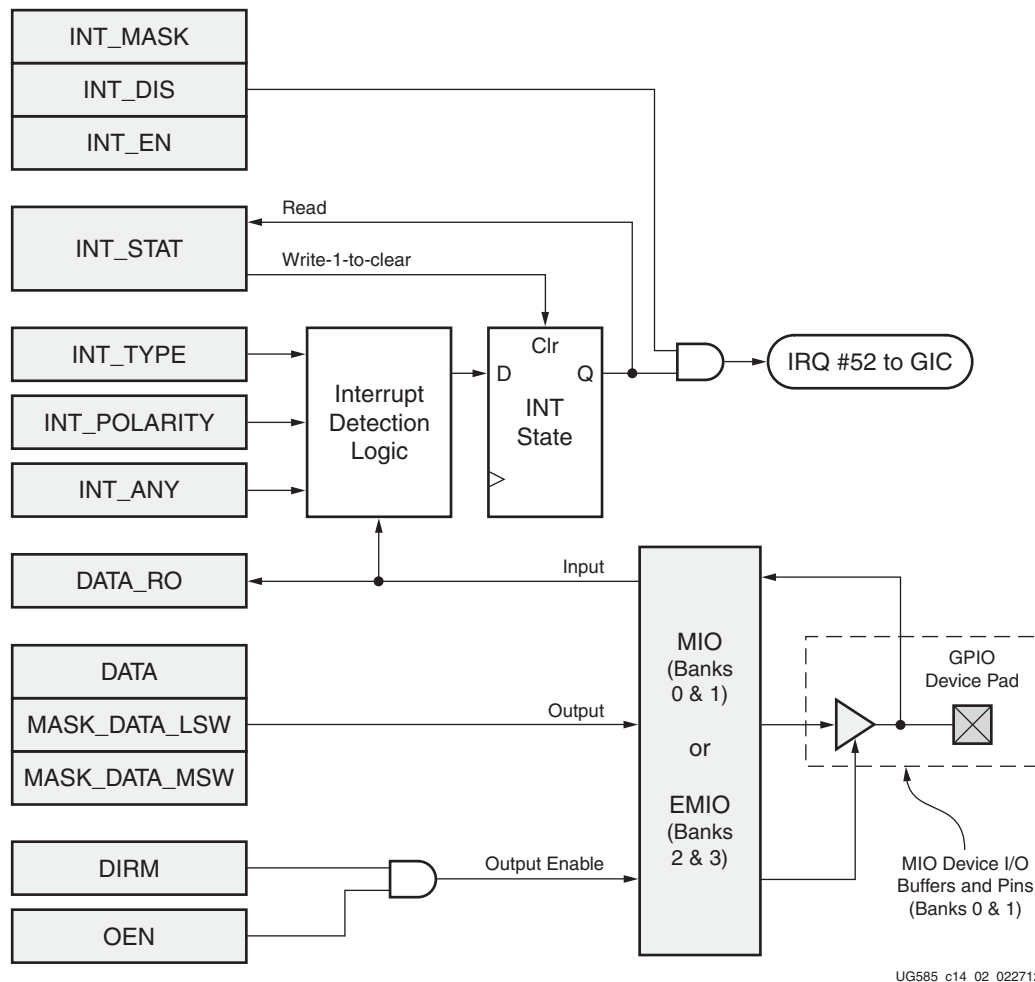


Figure 14-2: GPIO Channel

Software configures the GPIO as either an output or input. The DATA_RO register always returns the state of the GPIO pin regardless of whether the GPIO is set to input (OE signal false) or output (OE signal true). To generate an output waveform, software repeatedly writes to one or more GPIOs (usually using the MASK_DATA register).

Applications might need to switch more than one GPIO at the same time (less a small amount of inherent skew time between two I/O buffers). In this case, all of the GPIOs that need to be switched simultaneously must be from the same 16-bit half-bank (i.e., either the most-significant 16 bits or the least-significant 16 bits) of GPIOs to enable the MASK_DATA register to write to them in one store instruction.

MIO bank control (for Bank0 and Bank1) is summarized as follows:

- **DATA_RO:** This register enables software to observe the value on the device pin. If the GPIO signal is configured as an output, then this would normally reflect the value being driven on the output. Writes to this register are ignored.
Note: If the MIO is not configured to enable this pin as a GPIO pin, then DATA_RO is unpredictable because software cannot observe values on non-GPIO pins through the GPIO registers.
- **DATA:** This register controls the value to be output when the GPIO signal is configured as an output. All 32 bits of this register are written at one time. Reading from this register returns the previous value written to either DATA or MASK_DATA_{LSW,MSW}; it does not return the current value on the device pin.
- **MASK_DATA_LSW:** This register enables more selective changes to the desired output value. Any combination of up to 16 bits can be written. Those bits that are not written are unchanged and hold their previous value. Reading from this register returns the previous value written to either DATA or MASK_DATA_{LSW,MSW}; it does not return the current value on the device pin. This register avoids the need for a read-modify-write sequence for unchanged bits.
- **MASK_DATA_MSW:** This register is the same as MASK_DATA_LSW, except it controls the upper 16 bits of the bank.
- **DIRM:** Direction Mode. This controls whether the I/O pin is acting as an input or an output. Since the input logic is always enabled, this effectively enables/disables the output driver. When DIRM[x]=0, the output driver is disabled.
- **OEN:** Output Enable. When the I/O is configured as an output, this controls whether the output is enabled or not. When the output is disabled, the pin is 3-stated. When OEN[x]=0, the output driver is disabled.

14.3.1 Special Consideration for EMIO Signals

This section describes the operation of Bank2 and Bank3 (see [Figure 14-2](#)).

The register interface for the EMIO banks is the same as for the MIO banks described in the previous section. However, the EMIO interface is simply wires between the PS and the PL, so there are a few differences:

- The inputs are wires from the PL and are unrelated to the output values or the DIRM/OEN register. They can be read from the DATA_RO register.
- The output wires are not 3-state capable, so they are unaffected by DIRM/OEN as well. The value to be output is programmed using the DATA, MASK_DATA_LSW, and MASK_DATA_MSW registers.
- The output enable wires are simply outputs from the PS. These are controlled by the DIRM/OEN registers as follows: $EMIOGPOTN[x] = DIRM[x] \& OEN[x]$

The EMIO I/Os are not connected to the MIO I/Os in any way. The EMIO inputs cannot be connected to the MIO outputs and the MIO inputs cannot be connected to the EMIO outputs. Each bank is independent and can only be used as software observable/controllable signals.

14.3.2 Special Treatment of Bank0[8:7]

GPIO bits[8:7] of Bank0 correspond to package pins that are used to control the voltage mode of the I/O buffers themselves during reset. Therefore, they must be driven by the external system according to the proper voltage mode. In order to prevent them from being driven by other system logic, they cannot be used as general purpose inputs.

These bits can be used as general purpose outputs since the output is disabled at reset. The system can start using these as outputs after the voltage mode has been read during system boot.

14.4 Interrupt Function

The interrupt detection logic monitors the GPIO input signal. The interrupt trigger can be a positive edge, negative edge, either edge, Low-level or High-level. The trigger sensitivity is programmed using the INT_TYPE, INT_POLARITY and INT_ANY registers.

If an interrupt is detected, the GPIO's INT_STAT state is set true by the interrupt detection logic. If the INT_STAT state is enabled (unmasked), then the interrupt propagates through to a large OR function. This function combines all interrupts for all GPIOs in all four banks to one output (IRQ ID#52) to the interrupt controller. If the interrupt is disabled (masked), then the INT_STAT state is maintained until cleared, but it does not propagate to the interrupt controller unless the INT_EN is later written to disable the mask. As all GPIOs share the same interrupt, software must consider both INT_MASK and INT_STAT to determine which GPIO is causing an interrupt.

The interrupt mask state is controlled by writing a 1 to the INT_EN and INT_DIS registers. Writing a 1 to the INT_EN register disables the mask allowing an active interrupt to propagate to the interrupt controller. Writing a 1 to the INT_DIS register enables the mask. The state of the interrupt mask can be read using the INT_MASK register.

If the GPIO interrupt is edge sensitive, then the INT state is latched by the detection logic. The INT latch is cleared by writing a 1 to the INT_STAT register. For level-sensitive interrupts, the source of the interrupt input to the GPIO must be cleared in order to clear the interrupt signal. Alternatively, software can mask that input using the INT_DIS register.

The state of the interrupt signal going to the interrupt controller can be inferred by reading the INT_STAT and INT_MASK registers. This interrupt signal is asserted if INT_STAT=1 and INT_MASK=0.

MIO bank control (for Bank2 and Bank3) is summarized as follows:

- **INT_MASK:** This register is read-only and shows which bits are currently masked and which are un-masked/enabled.
- **INT_EN:** Writing a 1 to any bit of this register enables/unmasks that signal for interrupts. Reading from this register returns an unpredictable value.
- **INT_DIS:** Writing a 1 to any bit of this register masks that signal for interrupts. Reading from this register returns an unpredictable value.
- **INT_STAT:** This registers shows if an interrupt event has occurred or not. Writing a 1 to a bit in this register clears the interrupt status for that bit. Writing a 0 to a bit in this register is ignored.

- **INT_TYPE:** This register controls whether the interrupt is edge sensitive or level sensitive.
- **INT_POLARITY:** This register controls whether the interrupt is active-Low or active High (or falling-edge sensitive or rising-edge sensitive).
- **INT_ON_ANY:** If INT_TYPE is set to edge sensitive, then this register enables an interrupt event on both rising and falling edges. This register is ignored if INT_TYPE is set to level sensitive.

USB Host, Device, and OTG Controllers

15.1 Introduction

The Zynq-7000 PS includes two similar USB 2.0 controllers. Each controller is configured and controlled independently as Host, Device or OTG. The USB controllers (USB_x, where x is 0 or 1) use the ULPI protocol and can connect to either external PHYs or logic in the PL.

USB is a cable bus that supports data exchange between a host device and a wide range of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host-scheduled, token-based protocol. The bus allows peripherals to be attached, configured, used, and detached while the host and other peripherals are in operation.

The USB 2.0 specification supersedes the earlier versions of the USB specification. USB 2.0 offers the user a larger bandwidth increasing data throughput by a factor of 40. In addition to the 1.5 Mb/s and 12 Mb/s data rates of USB 1.1, the evolution to USB 2.0 adds an additional data rate of 480 Mb/s.

The OTG Supplement to the USB Specification extends USB to peer-to-peer applications. Using USB OTG technology, consumer electronics, peripherals, and portable devices can connect to each other. With USB On-The-Go, USB compatible peripheral devices can be developed that can also assume the role of a USB host.

15.1.1 Key Features

The two USB OTG peripherals have the following key features:

- USB 2.0 High Speed On-The-Go (OTG) dual-role USB Host controller or USB Device controller operation using the same hardware.
- USB 2.0 High Speed Device.
- USB 2.0 High Speed Host Controller.
- Intel® EHCI host controller. The USB Host controller registers and data structures are compatible to the Intel EHCI specification. Device controller registers and data structures are implemented as extensions to the EHCI programmers interface.
- Direct support for USB Transceiver Low Pin Interface (ULPI). The ULPI module supports 8 bits.
- Up to 12 endpoints.

The USB-HS controller features include:

- Directly connected USB legacy (USB 1.1). Full and low speed devices without a companion USB 1.1 Host controller or host controller driver software using EHCI standard data structures.
- Integrated transaction translator (multi port implementations) supports directly connected USB legacy (USB 1.1) Full and Low speed devices without a companion USB 1.1 Host controller or host controller driver software.
- Configurable dual port RAM buffers isolate memory latency on the system bus from the timing requirements of the USB.

15.1.2 System Viewpoint

The system viewpoint of the USB OTG is shown in [Figure 15-1](#).

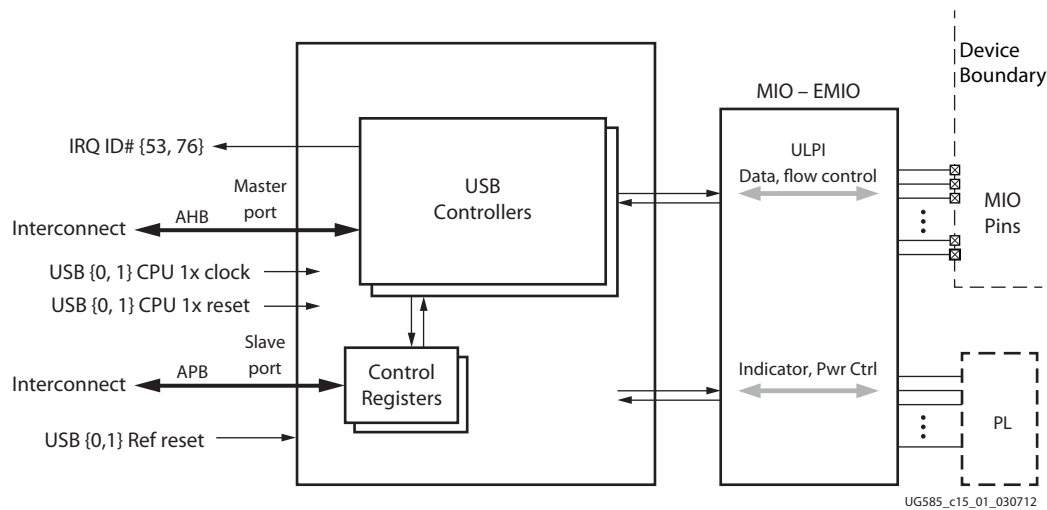


Figure 15-1: System Viewpoint Diagram

15.1.3 Notices

Low Power Mode

Low power mode is not supported.

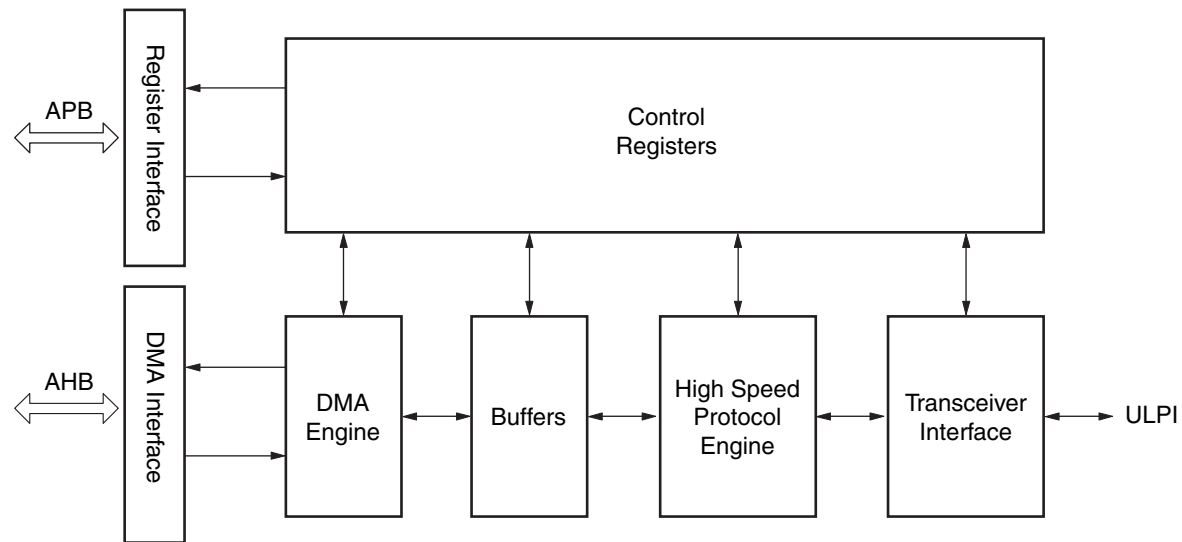
7x010 CLG225 Device

The 7z010 CLG225 device supports 32 MIO pins as shown in the MIO table in section [2.4.4 MIO-at-a-Glance Table](#). Only one USB interface is available in the 7z010 CLG222 device. If USB and GigE are required, the GigE signals must interface through the EMIO.

15.2 Functional Description

15.2.1 Block Diagram

The block diagram of the USB OTG is shown in [Figure 15-2](#).



UG585_c15_02_072512

Figure 15-2: USB OTG Block Diagram

15.2.2 Control Registers

The memory mapped status and control registers are accessed by software via an APB interface. Each controller includes its own set of registers. The register memory map is shown in [Table 4-6, page 92](#). Software is able to read the controller's capabilities, configure the controller, and control its operation.

Two groups of registers exist in the controller. The USB host controller registers are compatible with the USB host controller registers defined in the Intel EHCI specification. An additional set of registers provide control as a target device.

15.2.3 DMA Engine

The DMA engine block presents a bus master to the system interconnect. It is responsible for moving all of the data to be transferred over USB between the USB controller and buffers in system memory. The DMA controller must access both control information and packet data from system memory.

The control information is contained in link list based queue structures. The DMA controller has state machines that are able to parse all of the data structures defined in this controller specification.

- In host mode, the data structures are from the EHCI.
- In device mode, the data structures, designed to be similar to those in the EHCI specification, are used to allow device responses to be queued for each of the active pipes in the device.

15.2.4 Data Buffers

The data buffers are built as configurable FIFOs between the protocol engine block and the DMA controller. These FIFOs decouple the system memory bus request from the extremely tight timing required by the USB itself. 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 dual port memory.
- In device mode, multiple FIFO channels are maintained for each of the active endpoints in the system.

15.2.5 Protocol Engine

This block parses all the USB tokens and generates the response packets. It is responsible for:

- All error checking
- Field generation checking
- Formatting of all the necessary handshake actions
- Ping and data response packets on the bus

For any signal that must be generated based on a USB based time frame in host mode, the protocol engine also generates all of the token packets required by the USB protocol.

There is no separate transaction translator with the protocol engine. The transaction translator function associated with a USB 2.0 high speed hub is implemented within the DMA and the protocol engine blocks to better support connection to full and low speed devices.

15.2.6 Transceiver Interface

The primary function of this block is to isolate the rest of the USB peripheral from the transceiver, and to move all of the transceiver signaling into the primary clock domain of the peripheral. This allows the USB peripheral to run synchronously with the system processor and its associated resources.

The transceiver interface block interfaces to the 8-bit ULPI compatible external transceiver components.

15.2.7 USB Clocks

The USB ULPI clock is always an input to the Zynq-7000 EPP device.

15.3 Technology Description

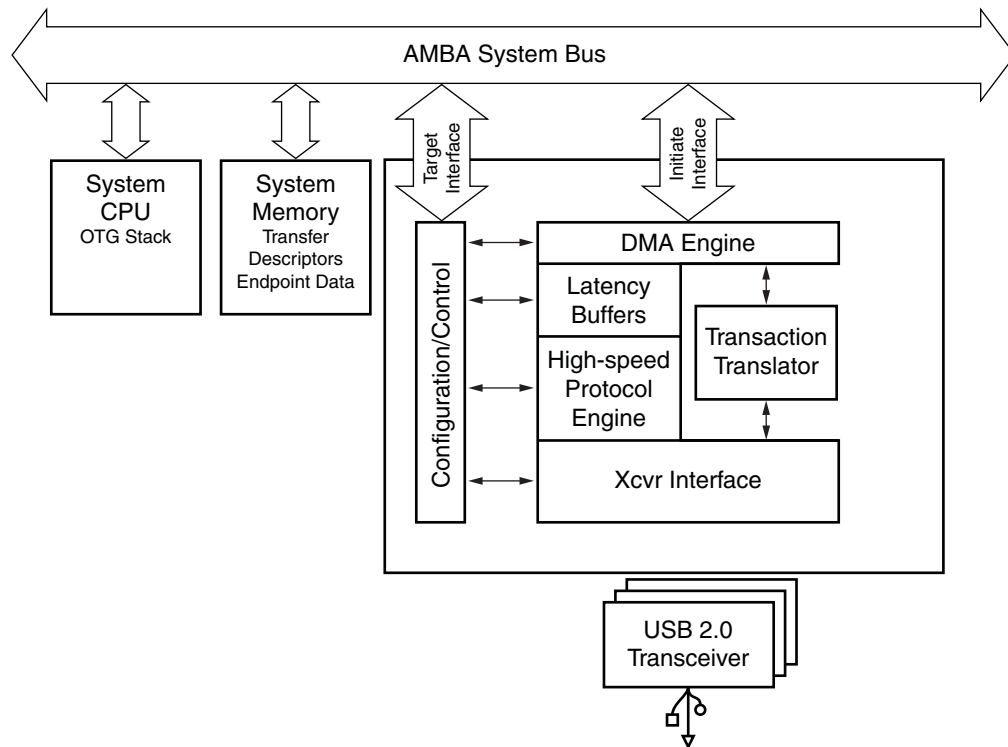
It is assumed that the reader has an understanding of USB technology. For additional information, please consult the *USB 2.0 Specification*, the *Enhanced Host Controller Interface Specification for Universal Serial Bus*, and the *AMBA™ Specification* for information on AMBA-AHB.

The USB-HS controller is designed to make efficient use of the system resources in an SOC design. The 32-bit system bus interface contains a chaining DMA engine that reduces the interrupt load on the application processor, and reduces the total system bus bandwidth that must be dedicated to servicing the USB interface requirements. By transferring the data to system memory at wire rates, the buffer memory requirement within the controller is minimized.

The USB-HS also makes strategic use of the processor for tasks that do not require timing critical responses to reduce the amount of special purpose logic.

15.3.1 Block Diagram

Figure 15-3 shows multiple transceivers, however PS has only one implemented per USB interface.



UG585_c15_03_042512

Figure 15-3: USB-HS Mode Block Diagram

15.3.2 Software Model

The device API provides a framework of routines to control the USB-HS peripheral in USB device applications.

The USB-HS device API is designed to significantly simplify the software tasks required to develop a USB device application. The API presents a high-level data transfer interface to the user's application code. All the register, interrupt and DMA interactions with the USB-HS controller are managed by the API. The API also includes routines that handle all the USB device framework commands which are required for all USB devices.

The host stack provides a layered software architecture to control all aspects of a USB bus system. The Host controller device (HCD) interface controls the functions of an embedded EHCI host controller. The USB driver layer provides all the USB driver functions to enumerate, manage and schedule a USB bus system, while the upper layers of the stack support standard USB device class interfaces to the device drives running on the embedded system.

Xilinx does not provide host stack, this should be provided by chosen operating system or third party vendor.

15.4 Device Data Structure

The function of the device operation is to transfer 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 will perform the data transfers.

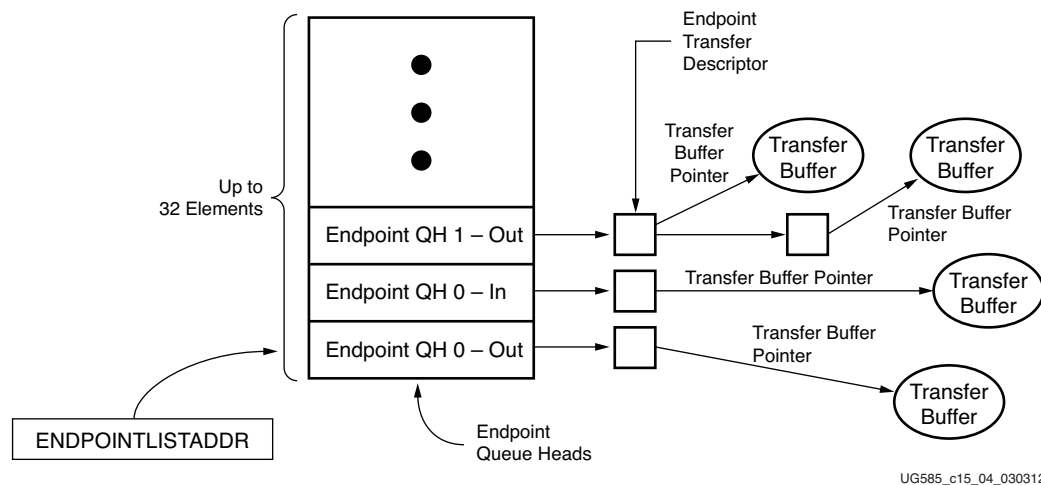


Figure 15-4: End Point Queue Head Organization

The USB-HS Device API incorporates and abstracts for the application developer all of the information contained in the device operational model. In this implementation of the USB Controller, all Endpoints for Device operation are bi-directional.

15.5 Host Data Structure

The host data structures are used to communicate control, status, and data between software and the Host controller. The periodic frame list is an array of pointers for the periodic schedule. A sliding window on the periodic frame List is used. The asynchronous transfer list is where all the control and bulk transfers are managed.

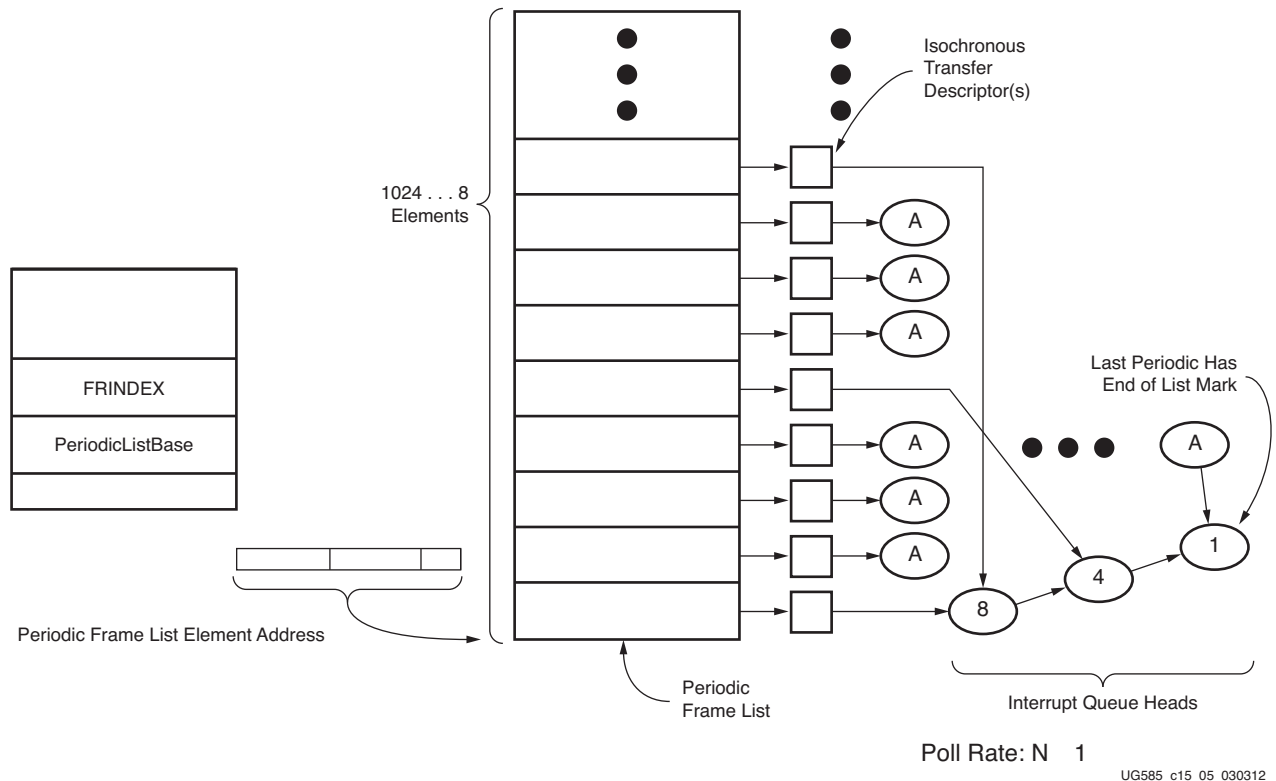


Figure 15-5: Periodic Schedule Organization

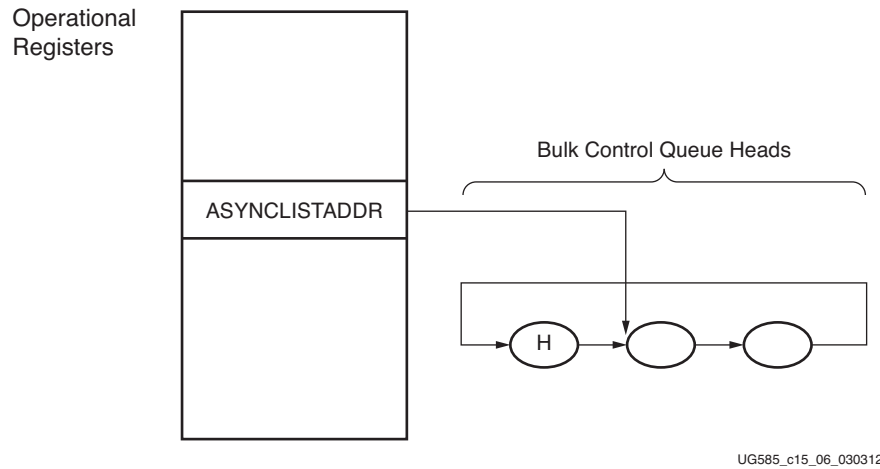


Figure 15-6: Asynchronous Schedule Organization

15.6 Hardware Model

15.6.1 USB-HS Block Diagram

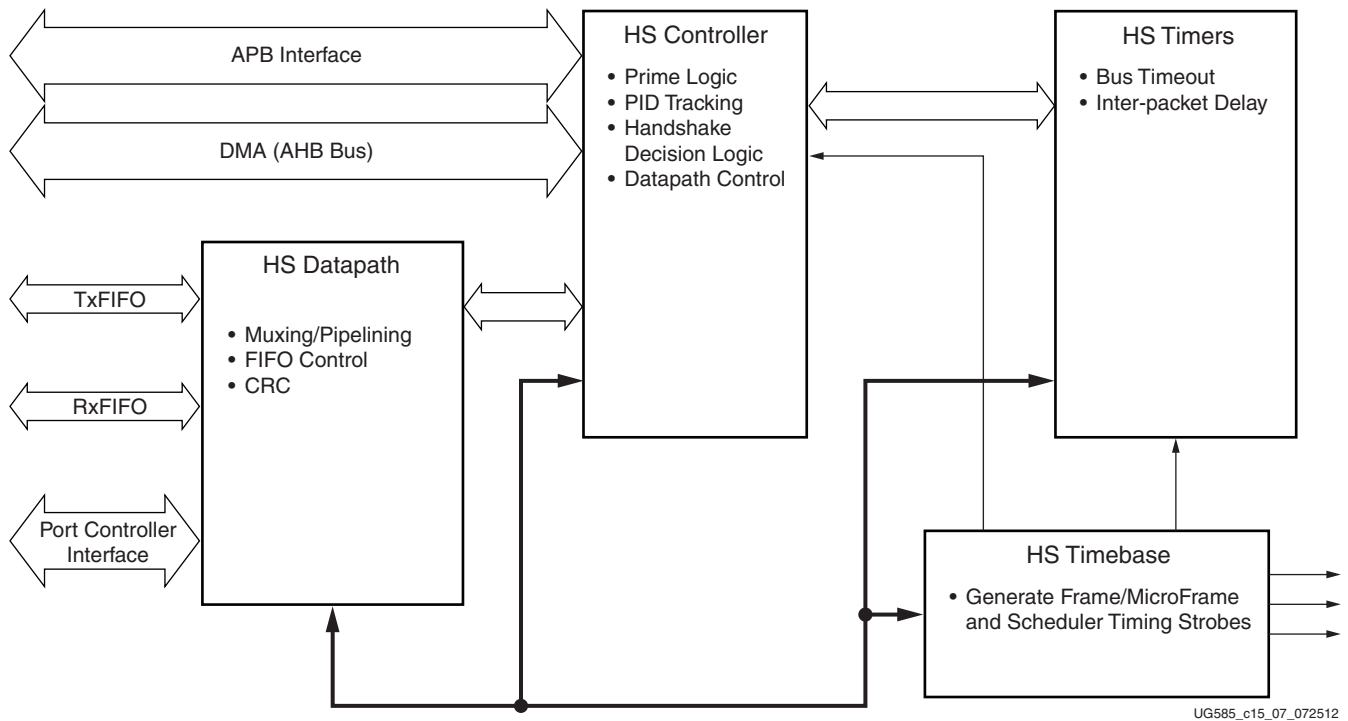


Figure 15-7: USB_HS Device, Host and OTG Diagram

15.6.2 DMA Engine

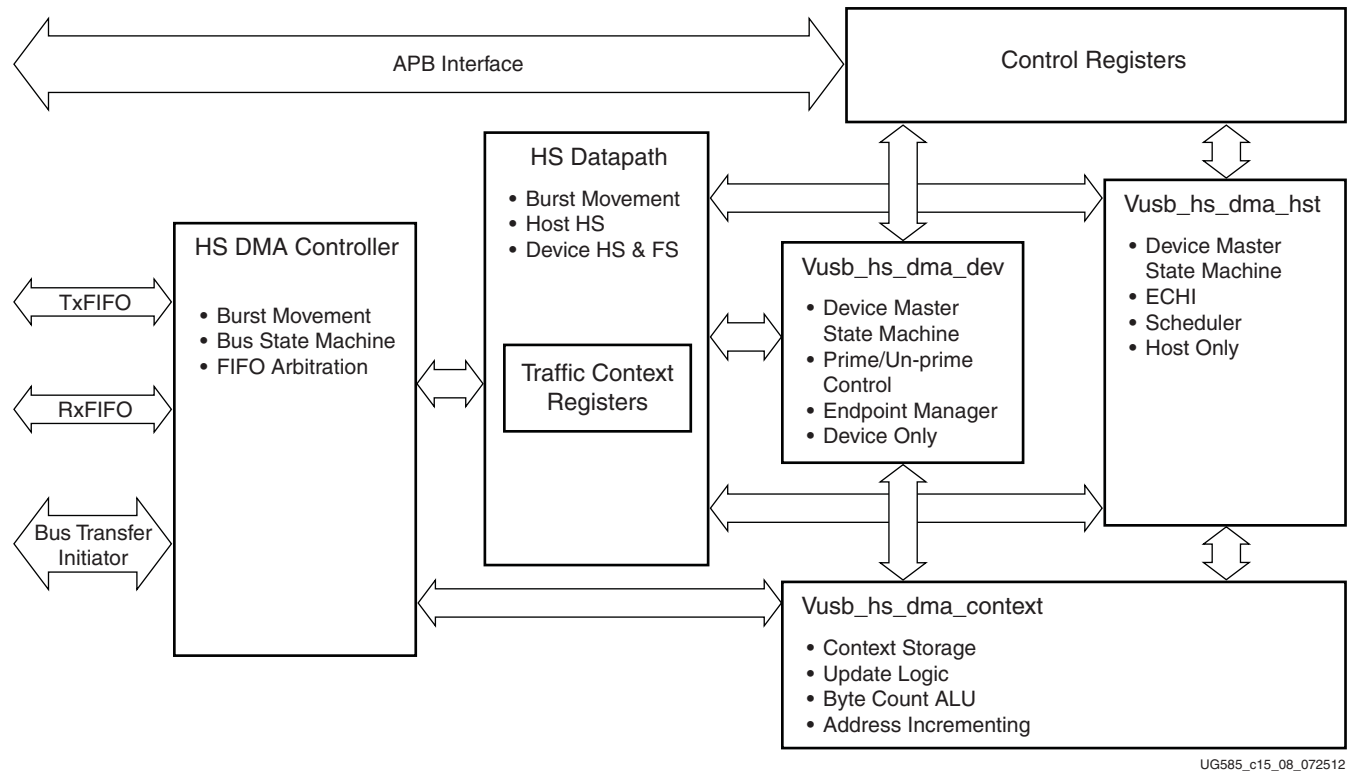


Figure 15-8: DMA Engine Block Diagram - OTG (Device/Host) Implementation

The DMA engine block presents a bus initiator (master) interface to the system memory bus. It is responsible for moving all of the data to be transferred over the USB between the USB-HS controller and buffers in system memory. Like the microprocessor interface block the DMA block uses a simple synchronous bus signaling protocol that eases connection to a number of different standard buses.

The DMA controller must access both control information and packet data from system memory. The control information is contained in link list based queue structures. The DMA controller has state machines that are able to parse all of the data structures defined in this controller specification. In host mode, the data structures are from the EHCI specification and represent queues of transfers to be performed by the host controller, including the split transaction requests that allow an EHCI controller to direct packets to Low and Full speed devices. In device mode, the data structures designed to be similar to those in the EHCI specification are used to allow device responses to be queued for each of the active pipes in the device.

15.6.3 Dual Port RAM Controller

The dual port RAM controller is used for context information and to build configurable FIFOs between the protocol engine block and the DMA controller. These FIFOs decouple the system processor memory bus request from the extremely tight timing required by the ULPI interface.

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 dual port memory. In Device mode, multiple FIFO channels are maintained for each of the active endpoints in the system.

15.6.4 Protocol Engine

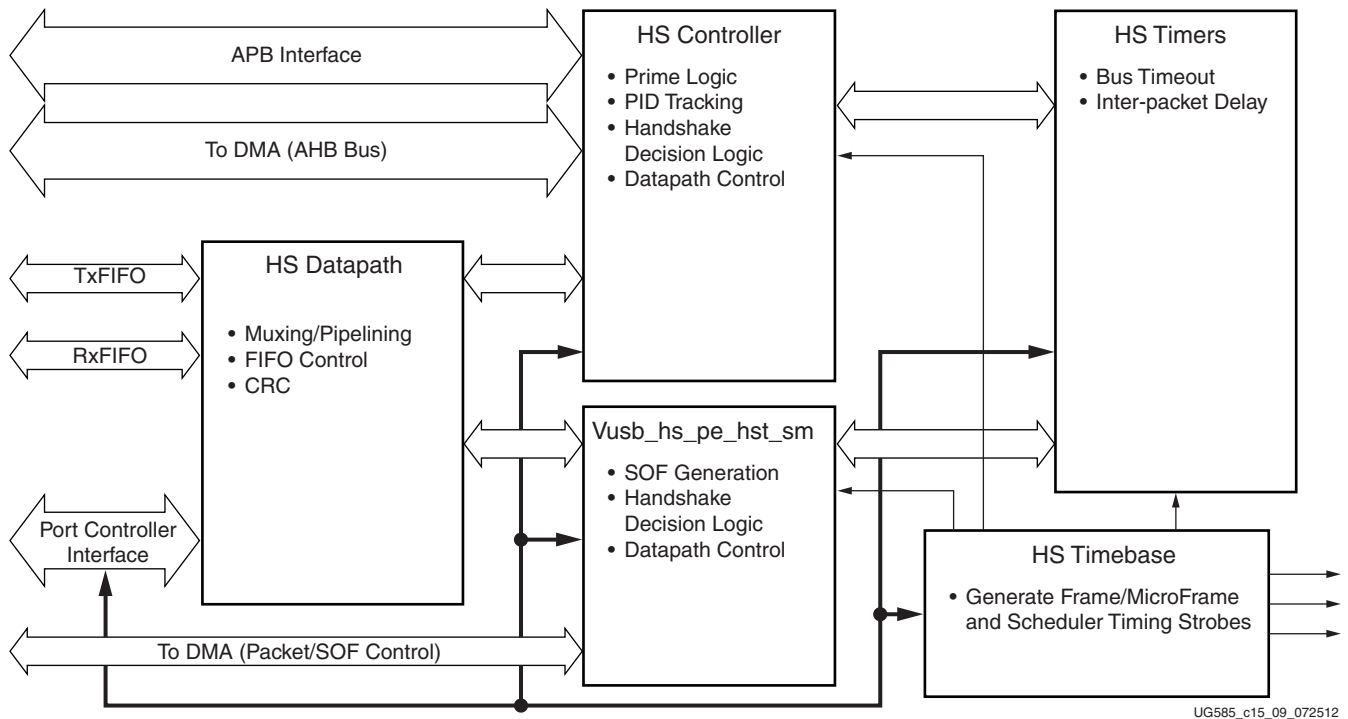


Figure 15-9: Protocol Engine Block Diagram - OTG (Device/Host) Implementation

The protocol engine parses all the USB tokens and generates the response packets. It is responsible for all error checking, check field generation, formatting all the handshake, ping and data response packets on the bus, and for any signals that must be generated based on a USB based time frame. In Host mode, the Protocol engine also generates all of the token packets required by the USB protocol. The Protocol engine contains several sub-functions:

- The token state machines track all of the tokens on the bus and filter the traffic based on the address and endpoint information in the token. In Host mode, these state machines also generate the tokens required for data transfer and bus control.
- The CRC5 and CRC16 CRC generator/checker circuits check and generate the CRC check fields for the token and data packets.
- The data and handshake state machines generate any responses required on the USB and move the packet data through the dual port memory FIFOs to the DMA controller block.
- The Interval timers provide timing strobes that identify important bus timing events: the bus timeout interval, the microframe interval, the start of frame interval, and the bus reset, resume, and suspend intervals.
- Reports all transfer status to the DMA engine.

15.6.5 Port Controller

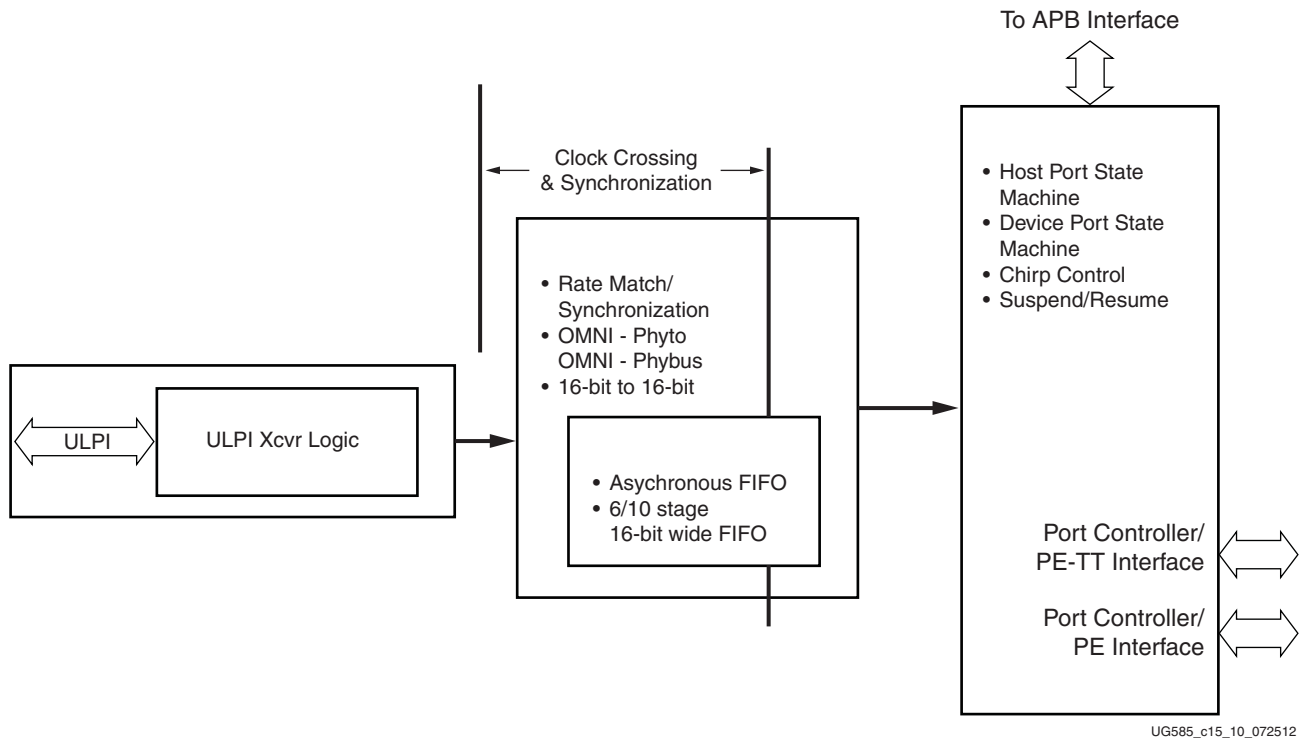


Figure 15-10: Port Controller Block Diagram

The port controller block interfaces to ULPI compatible transceiver macrocell core or component. The ULPI block shown in Figure 15-10 is an interface block to support ULPI external transceivers.

The primary function of the port controller block is to isolate the rest of the USB-HS controller from the transceiver, and to move all of the transceiver signaling into the primary clock domain of the USB-HS controller. This allows the controller to run synchronously with the system processor and its associated resources.

15.7 Operational Models

15.7.1 Host Operational Model

The USB Host controller follows the operational model as defined in the EHCI 1.0 specification.

For more details the EHCI 1.0 specification can be downloaded from:

<http://www.intel.com/technology/usb/ehcispec.htm>

USB Host controller specific EHCI deviations and enhancements are described in the following sections.

15.7.2 EHCI Deviation — Host Mode Only

The host mode operation of the controller is near EHCI compatible with few minor differences documented in this section.

The particulars of the deviations occur in the areas summarized here:

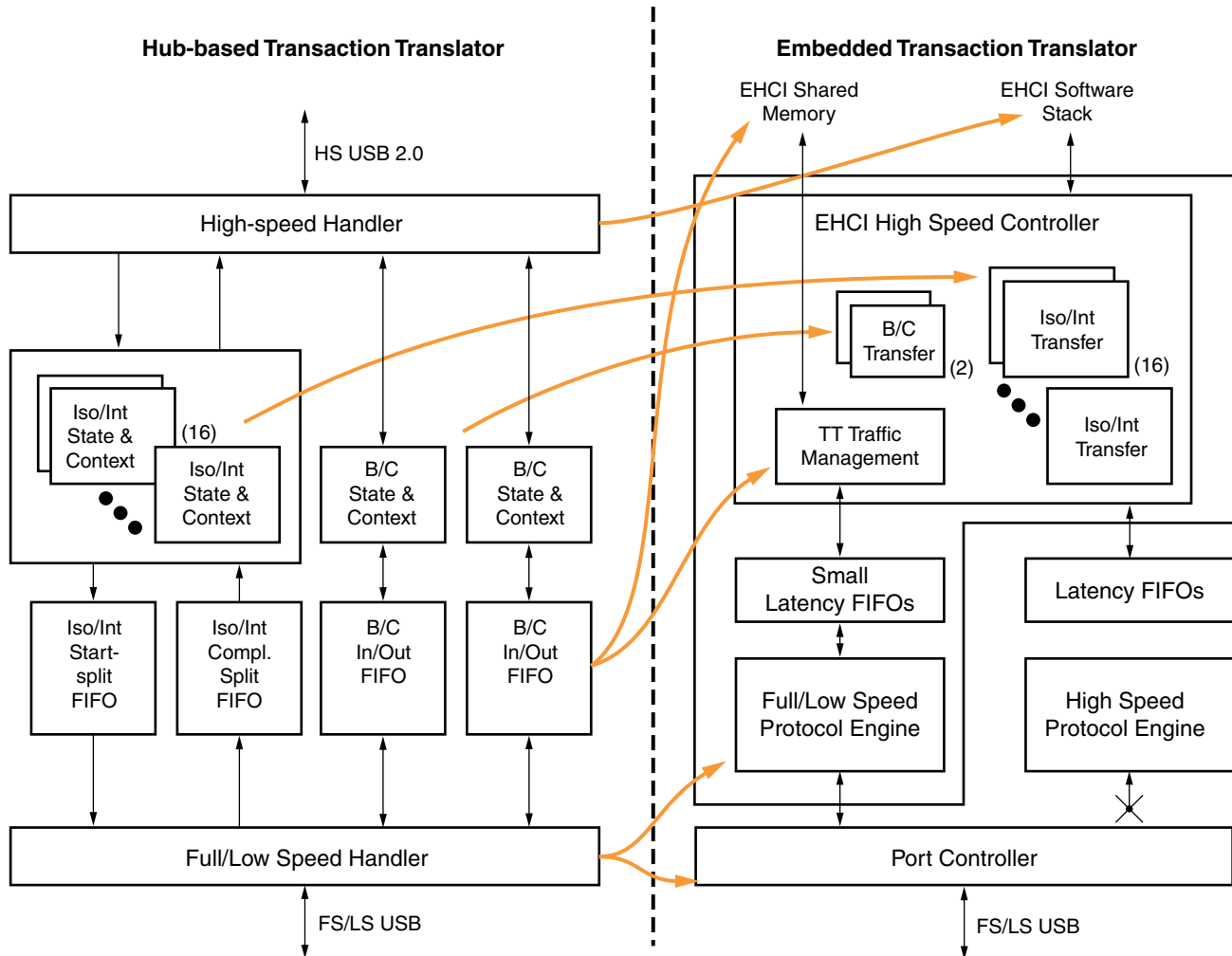
- Embedded transaction translator — Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.
- Embedded design interface — This controller does not have a PCI Interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.

15.7.3 Embedded Transaction Translator Function

The USB-HS controller working in Host mode supports directly connected Full and Low speed devices without requiring a companion controller by mimicking the capabilities of a USB 2.0 High speed hub transaction translator. Although there is no separate transaction translator block in the system, the transaction translator function normally associated with a high speed hub has been implemented within the DMA and protocol engine blocks.

The embedded transaction translator function is an extension to EHCI interface, but makes use of the standard data structures and operational models that exist in the EHCI specification.

15.7.4 Embedded Transaction Translator (Multi Port Host Only)



UG585_c15_11_072512

Figure 15-11: Hub vs. Embedded Representation of the Transaction Translator

On the left half of [Figure 15-11](#) the typical hub implementation of the transaction translator is shown with two ongoing asynchronous transactions capable of ping-pong access from each end. Periodic traffic is aggregated into a single data stream for each direction while a table of state and context for each pipe is stored within the hub-based transaction translator.

The right hand side of [Figure 15-11](#) shows how the same functions have been integrated into the host controller. The advantage of integrating those functions into the host controller is that the changes to the EHCI host controller driver are minimal while allowing direct connection of FS and LS devices without the need for a companion controller or external USB 2.0 hub. In addition, the host controller with the embedded transaction translator requires less local data storage than a hub-based transaction translator because the data storage is provided by main memory instead of hardware-based FIFOs.

If the size of each Iso/Int Transfer Context storage shown in the right side of [Figure 15-11](#) is significant, it is possible to reduce the number of these contexts from the default of 16 (required by the USB 2.0 specification) to 4. This sort of reduction can be carried out should it be determined that the host application is limited in scope in terms of host periodic schedule requirements.

Capability Registers

The following additions have been added to the capability registers to support the embedded transaction translator function:

- N_TT added to HCSPARAMS
- N_PTT added to HCSPARAMS

See HCSPARAMS for usage information.

Operational Registers

The following additions have been added to the operational registers to support the embedded TT:

- TTCTRL is a new register.
- Addition of two-bit Port Speed (PSPD) to the PORTSCx register.

Discovery

In a standard EHCI controller design, the EHCI host controller driver detects a Full speed (FS) or Low speed (LS) device by noting if the port enable bit is set after the port reset operation. The port enable is only be set in a standard EHCI controller implementation after the port reset operation and when the host and device negotiate a High-Speed connection (i.e., chirp completes successfully).

Because this controller has an embedded transaction translator, the port enable is always set after the port reset operation regardless of the result of the host device chirp result and the resulting port speed is indicated by the PSPD field in PORTSCx.

Therefore, the standard EHCI host controller driver requires an alteration to handle directly connected Full and Low speed devices or hubs.

The change is a fundamental one in that is summarized in [Table 15-1](#).

Table 15-1: EHCI Functional Differences

Standard EHCI	EHCI with Embedded Transaction Translator
After port enable bit is set following a connection and reset sequence, the device/hub is assumed to be HS.	After port enable bit is set following a connection and reset sequence, the device/hub speed is noted from PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub thus, all port-level control is performed through the Hub Class to the nearest Hub.	FS and LS device can be either downstream from a HS hub or directly attached. When the FS/LS device is downstream from a HS hub, then port-level control is done using the Hub Class through the nearest Hub. When a FS/LS device is directly attached, then port-level control is accomplished using PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub with HubAddr=X. [where HubAddr > 0 and HubAddr is the address of the Hub where the bus transitions from HS to FS/LS (i.e. Split target hub)]	FS and LS device can be either downstream from a HS hub with HubAddr = X [HubAddr > 0] or directly attached [where HubAddr = TTHA {TTHA is programmable and defaults to 0} and HubAddr is the address of the Root Hub where the bus transitions from HS to FS/LS (i.e. Split target hub is the root hub)]

Data Structures

The same data structures used for FS/LS transactions through a HS hub are also used for transactions through the Root Hub with the embedded transaction translator. Here it is demonstrated how the hub address and endpoint speed fields should be set for directly attached FS/LS devices and hubs:

- QH (for direct attach FS/LS) - Async. (Bulk/Control Endpoints) Periodic (Interrupt)
 - Hub Address = TTHA (default TTHA = 0)
 - Transactions to direct attached device/hub.
- QH.EPS = Port Speed EHCI Functional Differences
 - Transactions to a device downstream from direct attached FS hub.
- QH.EPS = Downstream Device Speed
 - Maximum Packet Size must be less than or equal 64 or undefined behavior might result.
 - When QH.EPS = 01 (LS) and PORTSCx.PSPD = 00 (FS), a LS-pre-pid will be sent before the transmitting LS traffic.
- siTD (for direct attach FS) - Periodic (ISO Endpoint)
 - All FS ISO transactions:
- Hub Address = (default TTHA = 0)
- siTD.EPS = 00 (full speed)
 - Maximum Packet Size must less than or equal to 1023 or undefined behavior might result.

Note: FSTN data structures should only be used if the FS or LS device is downstream of a high speed hub, these data structures do not apply to FS or LS devices that are connected directly to a host port.

Operational Model

The operational models are well defined for the behavior of the transaction translator (see *USB 2.0 specification*) and for the EHCI controller moving packets between system memory and a USB-HS hub. Since the embedded transaction translator exists within the host controller there is no physical bus between EHCI host controller driver and the USB FS/LS bus. These sections briefly discuss the operational model for how the EHCI and transaction translator operational models are combined without the physical bus between. The following sections assume the reader is familiar with both the EHCI and USB 2.0 transaction translator operational models.

Micro-frame Pipeline

The EHCI operational model uses the concept of H-frames and B-frames to describe the pipeline between the Host (H) and the Bus (B). The embedded transaction translator shall use the same pipeline algorithms specified in the USB 2.0 specification for a Hub-based Transaction Translator.

All periodic transfers always begin at B-frame 0 (after SOF) and continue until the stored periodic transfers are complete. As an example of the micro-frame pipeline implemented in the embedded transaction translator, all periodic transfers that are tagged in EHCI to execute in H-frame 0 will be ready to execute on the bus in B-frame 0.

It is important to note that when programming the S-mask and C-masks in the EHCI data structures to schedule periodic transfers for the embedded transaction translator, the EHCI host controller driver must follow the same rules specified in EHCI for programming the S-mask and C-mask for downstream hub-based transaction translators.

Once periodic transfers are exhausted, any stored asynchronous transfer will be moved. Asynchronous transfers are opportunistic in that they shall execute whenever possible and their operation is not tied to Hframe and B-frame boundaries with the exception that an asynchronous transfer cannot babble through the SOF (start of B-frame 0.)

Split State Machines

The start and complete split operational model differs from EHCI slightly because there is no bus medium between the EHCI controller and the embedded transaction translator. Where a start or complete-split operation would occur by requesting the split to the HS hub, the start/complete split operation is simple an internal operation to the embedded transaction translator. [Table 15-2](#) summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 15-2: Handshake Emulation Conditions

Condition	Emulate TT Response
Start-Split: All asynchronous buffers full.	NAK
Start-Split: All periodic buffers full.	ERR
Start-Split: Success for start of async. transaction.	ACK
Start-Split: Start periodic transaction.	No Handshake (OK)
Complete-Split: Failed to find transaction in queue.	Bus Time Out

Table 15-2: Handshake Emulation Conditions (Cont'd)

Condition	Emulate TT Response
Complete-Split: Transaction in queue is busy.	NYET
Complete-Split: Transaction in queue is complete.	[Actual Handshake]

Asynchronous Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

USB 2.0 - 11.17.3	Sequencing is provided and a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.
USB 2.0 - 11.17.4	Transaction tracking for 2 data pipes.
USB 2.0 - 11.17.5	Clear_TT_Buffer capability provided through the use of the TTCTRL register.

Periodic Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded transaction translator:

USB 2.0 - 11.18.6.[1-2]	Abort of pending start-splits EOF (and not started in micro-frames 6) Idle for more than 4 micro-frames Abort of pending complete-splits EOF Idle for more than 4 micro-frames
USB 2.0 - 11.18.[7-8]	Transaction tracking for up to 16 data pipes.

Caution: Limiting the number of tracking pipes in the embedded -TT to four (4) will impose the restriction that no more than four periodic transactions (INTERRUPT/ISOCRONOUS) can be scheduled through the embedded-TT per frame. the number 16 was chosen in the USB specification because it is sufficient to ensure that the high-speed to full-speed periodic pipeline can remain full. keeping the pipeline full puts no constraint on the number of periodic transactions that can be scheduled in a frame and the only limit becomes the flight time of the packets on the bus.

Complete-split transaction searching.

Note: There is no data schedule mechanism for these transactions other than the micro-frame pipeline. The embedded TT assumes the number of packets scheduled in a frame does not exceed the frame duration (1 ms) or else undefined behavior might result.

Multiple Transaction Translators

The maximum number of embedded transaction translators that is currently supported is one as indicated by the N_TT field in the HCSPARAM register.

Non-Zero Values in EHCI Reserved Fields

Write operations to all EHCI reserved fields (some of which are device fields) with the operation registers should always be written to zero.

Read operations by the Host controller must properly mask EHCI reserved fields.

SOF Interrupt

This SOF Interrupt is a free running 125 us interrupt for host mode. EHCI does not specify this interrupt but it has been added for convenience and as a potential software time base. See USBSTS and USBINTR registers.

15.7.5 Embedded Design Interface

This is an Embedded USB Host controller as defined by the EHCI specification and thus does not implement the PCI configuration registers.

Frame Adjust Register

Given that the optional PCI configuration registers are not included in this implementation, there is no corresponding bit level timing adjustments like is provided by the frame adjust register in the PCI configuration registers. Starts of micro-frames are timed precisely to 125 us using the transceiver clock as a reference clock. i.e., 60 MHz transceiver clock for 8-bit physical interfaces and full-speed serial interfaces.

15.7.6 Miscellaneous Variations from EHCI

Programmable Physical Interface Behavior

This design supports multiple physical interfaces that can operate in differing modes when the controller is configured with software programmable physical interface modes. Software programmability allows the selection of the physical interface part during the board design phase instead of during the chip design phase. The control bits for selecting the physical interface operating mode have been added to the PORTSC1 register providing a capability that is not defined by EHCI.

Discovery

Port Reset

The port connect methods specified by EHCI require setting the port reset bit in the PORTSC1 register for a duration of 10 ms. Because of the complexity required to support the attachment of devices that are not high speed there are counters already present in the design that can count the 10 ms 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 PR field of PORTSC1 register) to the reset the device.
- Software shall write a 0 (to PR field of PORTSC1 register) to the reset the device after 10 ms.

This step, which is necessary in a standard EHCI design, can 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 that the device is now operational and at this point the port speed has been determined.

Port Speed Detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation which re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed 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 (PSPD) has been added to PORTSC1 to provide the current operating speed of the port to the host controller driver.
- A 1-bit High Speed indicator (HSP) has been added to PORTSC to signify that the port is in High-Speed vs. Full/Low Speed - This information is redundant with the 2-bit Port Speed indicator above.

Port Test Mode

Port Test Control mode behaves fully as described in EHCI.

15.8 Device Operational Model

The function of the device operation is to transfer 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 will perform 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.

15.8.1 Device Controller Initialization

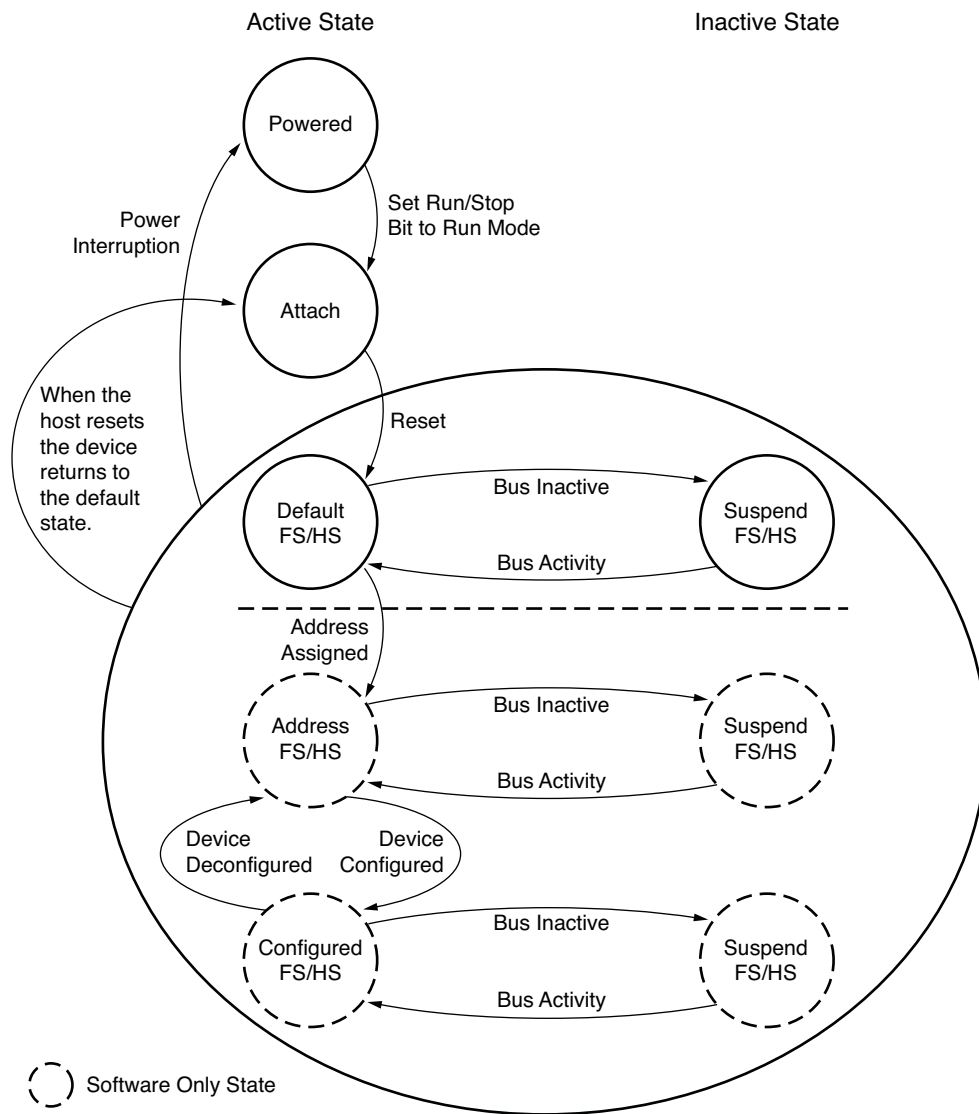
After hardware reset, the device is disabled until the Run/Stop bit is set to a 1. In the disabled state, the pull-up on the USB D+ is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs. 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 that the device controller can store the incoming setup packet.

In order to initialize a device, the software should perform the following steps:

1. Set controller mode in the USBMODE register to device mode.
 - Transitioning from host mode to device mode requires a device controller reset before modifying USBMODE.
 - Only needed to set when using OTG mode. In device-only mode, this is hardwired to device.
2. Allocate and initialize device queue heads in system memory.
 - Minimum: Initialize device queue heads 0 Tx and 0 Rx.
 - All device queue heads associated with control endpoints must be initialized before the control endpoint is enabled. Non-control device queue heads must be initialized before the endpoint is used and not necessarily before the endpoint is enabled.
 - For information on device queue heads, refer to [Device Data Structures](#).
3. Configure ENDPOINTLISTADDR pointer.
 - For information on ENDPOINTLISTADDR, refer to section Programmer's Manual document.
4. Enable the microprocessor interrupt associated with the USB-HS mode.
 - Enable interrupt signal in GIC (ID#53 for USB controller 0 and ID#76 for USB controller 1).
 - Recommended: enable all device interrupts including: USBINT, USBERRINT, port change detect, USB reset received, DCSuspend.
 - For a list of available interrupts refer to the USBINTR and the USBSTS register.
5. Set Run/Stop bit to Run Mode.
 - After the run bit is set, a device USB reset occurs. The DCD must monitor the reset event and adjust the software state as described in the Bus Reset section of the following Port State and Control section below.
 - Endpoint 0 is designed as a control endpoint only and does not need to be configured using ENDPTCTRL0 register.
 - It is also not necessary to initially prime Endpoint 0 because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework (Chapter 9) command set.

15.8.2 Port State and Control

From a chip or system reset, the device controller enters the powered state. A transition from the powered state to the attach state occurs when the Run/Stop bit is set to a 1. After receiving a reset on the bus, the port enters the default FS or default HS state in accordance with the reset protocol described in Appendix C.2 of the USB Specification Rev. 2.0. The state diagram shown in Figure 15-12 depicts the state of a USB 2.0 device.



UG585_c15_12_030312

Figure 15-12: Device Status Diagram

Table 15-3: Device Controller State Information Bits

Bit Description	Register Bit
DCSpend	USBSTS.SLI
USB Reset Received	USBSTS.URI

Table 15-3: Device Controller State Information Bits (Cont'd)

Bit Description	Register Bit
Port Change Detect	USBSTS.PCI
High-Speed Port	PORTSCx.PSPD

It is the responsibility of the DCD to maintain a state variable to differentiate between the default FS/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 device address register (DEVICEADDR) must be programmed by the DCD.

Entry into the configured state indicates that all endpoints to be used in the operation of the device have been properly initialized by programming the ENDPTCTRLx registers and initializing the associated queue heads.

Bus Reset

A bus reset is used by the host to initialize downstream devices. When a bus reset is detected, the device controller renegotiates its attachment speed, reset the device address to 0 and notify 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 any primed transactions are canceled by the device controller. The concept of priming will be clarified below, but the DCD must perform the following tasks when a reset is received.

Clear all setup token semaphores by reading the ENDPTSETUPSTAT register and writing the same value back to the ENDPTSETUPSTAT register. Clear all the endpoint complete status bits by reading the ENDPTCOMPLETE register and writing the same value back to the ENDPTCOMPLETE register.

Cancel all primed status by waiting until all bits in the ENDPTPRIME are 0 and then writing 0xFFFFFFFF to ENDPTFLUSH register.

Read the reset bit in the PORTSCx register and make sure that it is still active. A USB reset occurs for a minimum of 3 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).

A hardware reset can be performed by writing a one to the device controller reset bit in the USBCMD reset. A hardware reset will cause the device to detach from the bus by clearing the run/stop bit. Thus, the DCD must completely re-initialize the device controller after a hardware reset.

Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated.

At this time, the DCD might 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 PORTSCx 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 according to the *USB Chapter 9 - Device Framework*.

The device DCD can use the FS/HS mode information to determine the bandwidth mode of the device.

In some applications, it might 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.

Note: Before resume signaling can be used, the host must enable it by using the Set Feature command defined in device framework of the *USB 2.0 Specification*.

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 support 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*.

The USB-HS device controller hardware is configured to support up to 12 endpoints. The DCD can enable, disable and configure endpoint type up to the maximum selected during synthesis.

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, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a queue head allocated in memory. For 12 endpoints, numbers is used, then 24 queue heads are required one for each endpoint direction being used by the device controller. The operation of an endpoint and use of queue heads are described later in this document.

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 ENDPTCTRLx register. Each 32-bit ENDPTCTRLx is split into an upper and lower half. The lower half of ENDPTCTRLx is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the ENDPTCTRLx register otherwise the behavior is undefined. [Table 15-4](#) shows how to construct a configuration word for endpoint initialization.

Table 15-4: Device Controller Endpoint Initialization

Field	Value
Data Toggle Reset (ENDPTCTRLx.TXR)	1
Date Toggle Inhibit (ENDPTCTRLx.TXI)	0
Endpoint Type (ENDPTCTRLx.TXT)	00 — Control 01 — Isochronous 10 — Bulk 11 — Interrupt
Endpoint Stall (ENDPTCTRLx.TXS)	0

Stalling

There are two occasions where the device controller might need to return to the host a STALL. The first occasion is the functional stall, which is a condition set by the DCD as described in the USB 2.0 device framework. A functional stall is only used on non-control endpoints and can be enabled in the device controller by setting the endpoint stall bit in the ENDPTCTRLx register associated with the given endpoint and the given direction. In a functional stall condition, the device controller will continue 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, is used on control endpoints and 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 ENDPTCTRLx register can ensure that both stall bits are set at the same instant.

Note: Any write to the ENDPTCTRLx register during operational mode must preserve the endpoint type field (i.e. perform a read-modify-write).

Table 15-5: 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/NYE
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/NYE

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.

Data Toggle Reset

The DCD can 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 `ENDPTCTRLx.TXR` register. This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

Data Toggle Inhibit

This feature 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 device controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state.

In normal operation, the device 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 host controller from re-sending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response.

15.8.3 Operational Model for Packet Transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the *USB 2.0 Specification*. At USB 1.1 Full or Low Speed rates, this turnaround time was significant and the USB 1.1 device controllers were designed so that the device controller could access main memory or interrupt a host protocol processor in order to respond to the USB 1.1 transaction. The architecture of the USB 2.0 device controller must be different because same methods will not meet USB 2.0 High-speed turnaround time requirements by simply increasing clock rate.

A USB host sends requests to the device 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, then we can expect the host will send IN requests to that endpoint.

This device controller is designed in such a way that it can prepare packets for each endpoint or 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 will be used throughout the following documentation to describe the device controller operation so the DCD can be designed properly use priming. Further, note that the term “flushing” is used to describe the action of clearing a packet that was queued for execution.

Priming Transmit Endpoints

Priming a transmit endpoint will cause 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 will be 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 handle 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. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to the maximum number of endpoints configured at device synthesis time.

After a priming request is complete, an endpoint state of primed is indicated in the `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.

Since 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. More information about FIFO sizing is presented in section Bandwidth and Latency Issues.

Priming Receive Endpoints

Priming receive endpoints is 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 that there is no data movement of the leading packet data simply because the data is to be received from the host.

Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the `RxFIFO` does not scale with the number of endpoints.

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 will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence.

A dTD will be 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 describe 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,
$$N = \text{INT}(\text{Number of Bytes}/\text{Max. Packet Length}) + 1$$

With Zero Length Termination (ZLT) = 1,
$$N = \text{MAXINT}(\text{Number of Bytes}/\text{Max. Packet Length})$$

Table 15-6: Variable Length Transfer Protocol Example (ZLT = 0)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	256	
512	256	3	256	256	0
512	512	2	512	0	

Table 15-7: Variable Length Transfer Protocol Example (ZLT = 1)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	256	
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 follows 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 length packet 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 will be retired. When the device is receiving, if the last packet length received equal maximum packet length and the total bytes is zero, it will wait 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 will not append any zero length packet. When receiving, it will 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 just one dTD, so it does not make sense to add a zero length termination packet each time a dTD is consumed. On those cases we recommend 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 equals zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. Total bytes in dTD equals 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 will discard the remaining packet, and set the Buffer Error bit in the dTD. In addition, the endpoint will be flushed and the USBERR interrupt will become active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD will be cleared and the next pointer will be followed when the Terminate bit is clear. When the terminate bit is set, the device controller will flush the endpoint/direction and cease 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. In order to recover from this error condition, the DCD must properly reinitialize the dQH by clearing the active bit and update the next TD 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 handling such errors.

Interrupt/Bulk Endpoint Bus Response Matrix

Table 15-8: Interrupt/Bulk Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow	Not Enabled
Setup	Ignore	Ignore	Ignore	N/A	N/A	BTO
In	STALL	NAK	Transmit	BS Error	N/A	BTO
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK	BTO
Ping	STALL	NAK	ACK	N/A	N/A	BTO
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	BTO

Notes:

1. BS Error — Force Bit Stuff Error
2. NYET/ACK — NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK
3. SYSERR — System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive
4. BTO -bus time out

Control Endpoint Operation Model

Setup Phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The device controller always accepts the setup phase unless the setup lockout is engaged.

The setup lockout will engage so that 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 will ensure that the setup packet payload is extracted from the queue head without being corrupted by an incoming setup packet. This is the 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 field) in USBMODE. (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

After receiving an interrupt and inspecting ENDPTSETUPSTAT to determine that a setup packet was received on a particular pipe:

1. Write 1 to setup tripwire (SUTW) in USBCMD register.
2. Duplicate contents of dQH.SetupBuffer into local software byte array.
3. Read setup tripwire (SUTW) in USBCMD register.
 - a. If set, continue to [step 4](#)
 - a. If cleared, go to [step 1](#)
4. Write 1 to clear corresponding bit ENDPTSETUPSTAT.
5. Write 0 to clear setup tripwire (SUTW) in USBCMD register.
6. Process setup packet using local software byte array copy and execute status/handshake phases.
7. Before priming for status/handshake phases, ensure that ENDPTSETUPSTAT is 0.

A poll loop should be used to wait until ENDPTSETUPSTAT transitions to 0 after [step 4](#). above and before priming for the status/handshake phases.

The time from writing a 1 to ENDPTSETUPSTAT and reading back a 0 is very short (~1–2 μ s) so a poll loop in the DCD is not be harmful.

Note: After receiving a new setup packet, the status and/or handshake phases might still be pending from a previous control sequence. These should be flushed and deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

Data Phase

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 ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime completes when the associated bit in the ENDPTPRIME register is zero and the associated bit in the ENDPTSTATUS register is a one. If a prime fails, i.e., the ENDPTPRIME bit goes to zero and the 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 ENDPTPRIME bit is cleared, then 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 prime status (ENDPTSTATUS) 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. Error handling of data phase packets is the same as bulk packets described previously.

Status Phase

Similar to the data phase, the 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 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. Error handling of data phase packets is the same as bulk packets described previously.

Control Endpoint Bus Response Matrix

The device controller response to packets on a control endpoint according to the device controller state is shown in Table 15-9.

Table 15-9: Control Endpoint Bus Response Matrix

Token Type	Endpoint State						Setup Lockout
	Stall	Not Primed	Primed	Under-flow	Over-flow	Not Enabled	
Setup	ACK	ACK	ACK	N/A	SYSERR	BTO	
In	STALL	NAK	Transmit	BS Error	N/A	BTO	N/A
Out	STALL	NAK	Receive + NYET/ACK	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

Table 15-9: Control Endpoint Bus Response Matrix (Cont'd)

Token Type	Endpoint State						Setup Lockout
	Stall	Not Primed	Primed	Under-flow	Over-flow	Not Enabled	

Notes:

1. BS Error — Force Bit Stuff Error
2. NYET/ACK — NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK
3. SYSERR — System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive
4. BTO -bus time out

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 device controller is accomplished by the following:

- Exactly MULT Packets per (u)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 endpoints. 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 (u)frame. If the ISO-dTD is still active after that frame, then the ISO-dTD is held ready until executed or canceled by the DCD.

Note: If the MULT field is set to more packets than present in the dTD to be transmitted, the controller sends zero length packets to all extra incoming IN tokens and report fulfillment error (transaction error) in current dTD. If more dTDs exist in memory, the USB-HS controller moves to the next dTD to be transmitted in the next (u)frame. Because of this behavior it is recommended to always use the correct MULT matching the number of packets to be processed for a given dTD.

An EHCI compatible host controller 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 used for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is that priming an ISO-endpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit will be cleared as usual to indicate to software that 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 that the device controller can match the dTD to a specific (u)frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (u)frame. Once an ISO transaction is started in a (u)frame it will retire 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 will force retire the ISO-dTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

Finally, 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 (u)frame to (u)frame operation the DCD should ensure that the dTD linked-list is out ahead of the device controller by at least two (μ)frames.

Isochronous Pipe Synchronization

When it is necessary to synchronize an isochronous data pipe to the host, the (u)frame number (FRINDEX register) can be used as a marker. To cause a packet transfer to occur at a specific (u)frame number [N], the DCD should interrupt on SOF during frame N-1. When the FRINDEX=N-1, the DCD must write the prime bit. The device controller primes the isochronous endpoint in (u)frame N-1 so that the device controller executes delivery during (u)frame N.

Caution: Priming an endpoint towards the end of (u)frame N-1 does not guarantee delivery in (u)frame N. The delivery might actually occur in (u)frame N+1 if device controller does not have enough time to complete the prime before the SOF for packet N is received.

Isochronous Endpoint Bus Response Matrix

Table 15-10: Isochronous Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow	Not Enabled
Setup	STALL	STALL	STALL	N/A	N/A	BTO
In	NULL Packet	NULL Packet	Transmit	BS Error	N/A	BTO
Out	Ignore	Ignore	Receive	N/A	Drop Packet	BTO
Ping	Ignore	Ignore	Ignore	Ignore	Ignore	BTO
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	BTO

Notes:

1. BS Error — Force Bit Stuff Error
2. NULL Packet — Zero length packet
3. BTO — bus time out

15.8.4 Managing Queue Heads

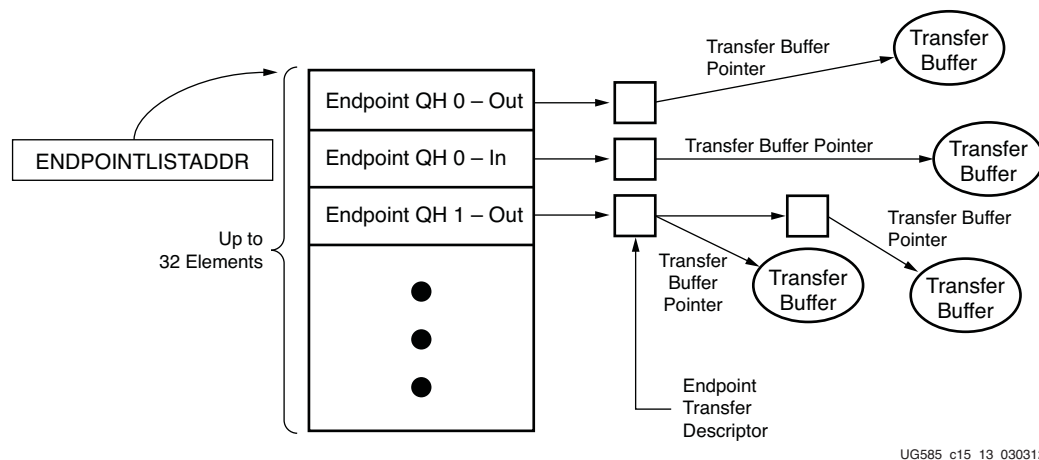


Figure 15-13: End Point Queue Head Diagram

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 ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown in Figure 15-13. The even elements in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTD has been retired, it will no longer be part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors since pointers will no longer exist within the queue head once the dTD is retired (see [Software Link Pointers](#)).

In addition to the current and next pointers and the dTD overlay examined in [Operational Model for Packet Transfers](#), the 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.

Queue Head Initialization

One pair of device queue heads must be initialized for each active endpoint. To initialize a device queue head:

1. Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
2. 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 and in conjunction with the USB Chapter 9 protocol. Note: In FS mode, the multiplier field can only be 1 for ISO endpoints.
3. Write the next dTD Terminate bit field to 1.
4. Write the Active bit in the status field to 0.
5. 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.

Operational Model for Setup Transfers

As discussed in [Control Endpoint Operation Model](#), setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8-byte buffer within the dQH.

Upon receiving notification of the setup packet, the DCD should handle the 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 ENDPTSETUPSTAT.
 - a. The acknowledge must occur before continuing to process the setup packet.
 - b. 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 dTD's from previous control transfers and flush if any exist as discussed in section Flushing/De-priming an Endpoint.
 - a. 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.

15.8.5 Managing Transfers with Transfer Descriptors

Software Link Pointers

It is necessary for the DCD software to maintain head and tail pointers to 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 next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list.

Note: To conserve memory, the reserved fields at the end of the dQH can be used to store the Head and Tail pointers but it still remains the responsibility of the DCD to maintain the pointers.

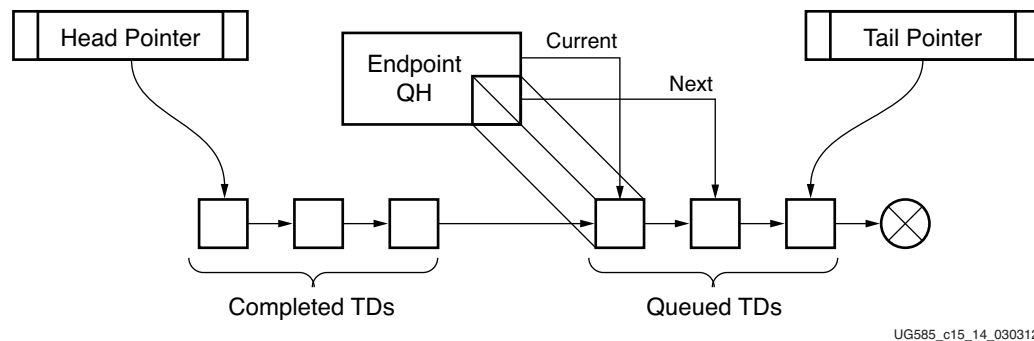


Figure 15-14: Software Link Pointers

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-DWORD dTD block of memory aligned to 8-DWORD boundaries. Example: bit address 4:0 would be equal to 00000.

Write the following fields:

1. Initialize first 7 DWORDs 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.

Executing a Transfer Descriptor

To safely add a dTD, the DCD must follow this procedure which will handle the event where the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

Determine whether the link list is empty:

1. 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

1. Write dQH next pointer AND dQH terminate bit to 0 as a single DWORD operation.
2. Clear active and halt bit in dQH (in case set from a previous error).
3. Prime endpoint by writing 1 to correct bit position in ENDPTPRIME.

Case 2: Link list is not empty

1. Add dTD to end of linked list.
2. Read correct prime bit in ENDPTPRIME - if 1 DONE.
3. Set ATDTW bit in USBCMD register to 1.
4. Read correct status bit in ENDPTPRIME. (store in tmp. variable for later)
5. Read ATDTW bit in USBCMD register.

If 0 go to [step 3](#).

If 1 continue to [step 6](#).

6. Write ATDTW bit in USBCMD register to 0.
7. If status bit read in (4) is 1 DONE.
8. If status bit read in (4) is 0 then Goto [Case 1: step 1](#).

Transfer Completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the host-initiated request. The DCD will be 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 that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following 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, the DCD must take proper action. Transfer failure mechanisms are indicated in the device error matrix.

In addition to checking the status bit the 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 reach zero, but for receive packets, the host might send fewer bytes in the transfer according the USB variable length packet protocol.

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 can also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

1. Write a 1 to the corresponding bit(s) in ENDPTFLUSH.
2. Wait until all bits in ENDPTFLUSH are 0.

Software note: This operation can 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 ENDPTSTAT to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now 0. If the corresponding bits are 1 after step #2 has finished, then the flush failed: in very rare cases, a packet is in progress to the particular endpoint when commanded flush using ENDPTFLUSH. A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD might need to repeatedly flush any endpoints that fail to flush be repeating steps 1–3 until each endpoint is successfully flushed.

Device Error Matrix

Table 15-11 summarizes packet errors that are not automatically handled by the device controller.

Table 15-11: Device Error Matrix

Error	Direction	Packet Type	Data Buffer Error Bit	Transaction Error Bit
Overflow	RX	Any	1	0
ISO Packet Error	RX	ISO	0	1
ISO Fulfillment Error	Both	ISO	0	1

Notes:

1. See Table 15-12.

Notice that the device controller handles 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 15-12: Device Error Descriptions

Error	Descriptions
Overflow	Number of bytes received exceeded max. packet size or total buffer length. This error will also set the Halt bit in the dQH and if there are dTDs remaining in the linked list for the endpoint, then those will not be executed.
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 (u)frame. For scheduled data delivery the DCD might need to readjust the data queue because a fulfillment error will cause Device Controller to cease data transfers on the pipe for one (μ)frame. During the “dead” (μ)frame, the Device Controller reports error on the pipe and primes for the following frame.

15.8.6 Servicing Interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

High-Frequency Interrupts

High frequency interrupts in particular should be handed in the order shown in [Table 15-13](#). The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

Table 15-13: High Frequency Interrupt Events

Execution Order	Interrupt	Action
1a	USB Interrupt ⁽¹⁾ ENDPTSETUPSTATUS	Copy contents of setup buffer and acknowledge setup packet (as indicated in Managing Queue Heads). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol.
1b	USB Interrupt ⁽¹⁾ ENDPTCOMPLETE	Handle completion of dTD as indicated in Managing Queue Heads .
2	SOF Interrupt	Action as deemed necessary by application. This interrupt might not have a use in all applications.

Notes:

1. It is likely that multiple interrupts to stack up on any call to the Interrupt Service Routine and during the execution of the interrupt service routine.

Low-Frequency Interrupts

The low frequency events include the following interrupts. These interrupt can be handled in any order since they do not occur often in comparison to the high-frequency interrupts.

Table 15-14: Low Frequency Interrupt Events

Interrupt	Action
Port Change	Change software state information.
Sleep Enable (Suspend)	Change software state information. Low power handling as necessary.
Rest Received	Change software state information. Abort pending transfers.

Error Interrupts

Error interrupts will be least frequent and should be placed last in the interrupt service routine.

Table 15-15: 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 will more aptly handle packet-level errors by checking dTD status field upon receipt of USB Interrupt (w/ ENDPTCOMPLETE).
System Error	Unrecoverable error. Immediate Reset of controller, free transfers buffers in progress and restart the DCD.

15.9 OTG Operations

15.9.1 Register Bits

In the previous sections, device mode and for host mode behaviors are described. However, during OTG operations it is necessary to perform tasks independent of the controller mode.

Note also from USBMODE register 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.

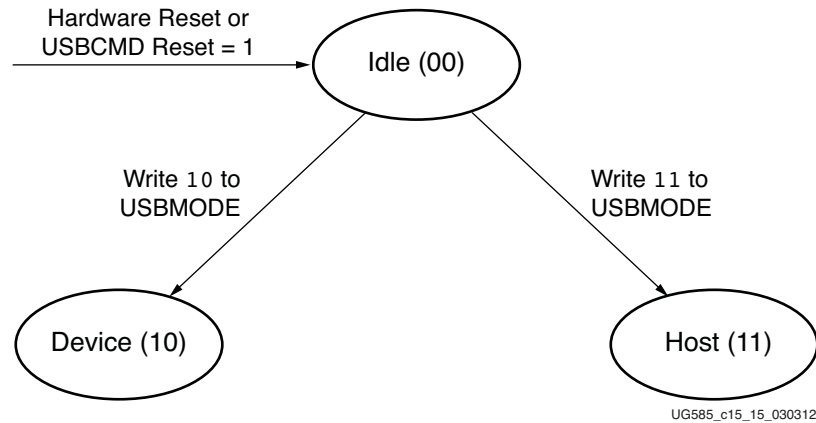


Figure 15-15: **Controller Mode**

To this end, the list below are the register bits that are 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 USBCMD register:

- All identification registers
- All device/host capability registers
- OTGSC: All bits
- PORTSC:
 - Physical interface select
 - Physical interface serial select
 - Physical interface data width
 - Physical interface low power
 - Physical interface wake signals
- Port indicators
- Port power

15.9.2 Hardware Assist

The hardware assist provides automated response and sequencing that might 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.

Auto-Reset

When the HAAR is set to one, the host will automatically start a reset after a connect event. This shortcuts the normal process where software is notified of the connect event and starts the reset. Software will still receive notification of the connect event but should not write the reset bit when the HAAR is set. Software will be notified again after the reset is complete via the enable change bit in the PORTSC register which cause a port change interrupt.

This assist will ensure the OTG parameter TB_ACON_BSE0_MAX = 1 ms is met.

Data-Pulse

Writing a one to HADP will start a data pulse of approximately 7 ms in duration and then automatically cease the data pulsing. During the data pulse, the DP will be 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 will ensure data pulsing meets the OTG requirement of > 5 ms and < 10 ms.

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 pull-up 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 controller.
2. Set the OTG controller into device mode.
3. Write the device run bit to a 1 and enable necessary interrupts including:
4. USB Reset Enable (URE); enables interrupt on USB bus reset to device
5. Sleep Enable (SLE); enables interrupt on device suspend
6. 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 control registers until it gets an interrupt from the device controller signifying that a reset interrupt has occurred or at least first verify that the controller has entered device mode. HCD/DCD must not activate the soft reset at any time since this action is performed by hardware. During the transition, the software might see an interrupt from the disconnect and/or other spurious interrupts (i.e., SOF/etc.) that might or might not cascade and can be cleared by the soft reset depending on the software response time.

After the controller 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 can be initiated quickly (a few microseconds) after connect, will require 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 whichever occurs first.

In the case where the A-peripheral fails to see a reset after the controller enters device mode and engages the DP-pull-up, the device controller interrupt the DCD signifying that a suspend has occurred.

This assist will ensure the parameter TA_BDIS_ACON_MAX = 3 ms is met.

15.10 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 chapter 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.

Note: The 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.

Note: The specification described in this chapter is based on *EHCI 1.0 Specification*. In case of ambiguity or conflicting statements between the description in this chapter and EHCI 1.0 description, *EHCI Specification* takes precedence.

15.10.1 Periodic Frame List

This schedule is for all periodic transfers (isochronous and interrupt). The periodic schedule is referenced from the operational registers space using the PERIODICLISTBASE address register and the FRINDEX register. The periodic schedule is based on an array of pointers called the Periodic Frame List. The PERIODICLISTBASE address register is combined with the FRINDEX register to produce a memory pointer into the frame list. The Periodic Frame List implements a sliding window of work over time.

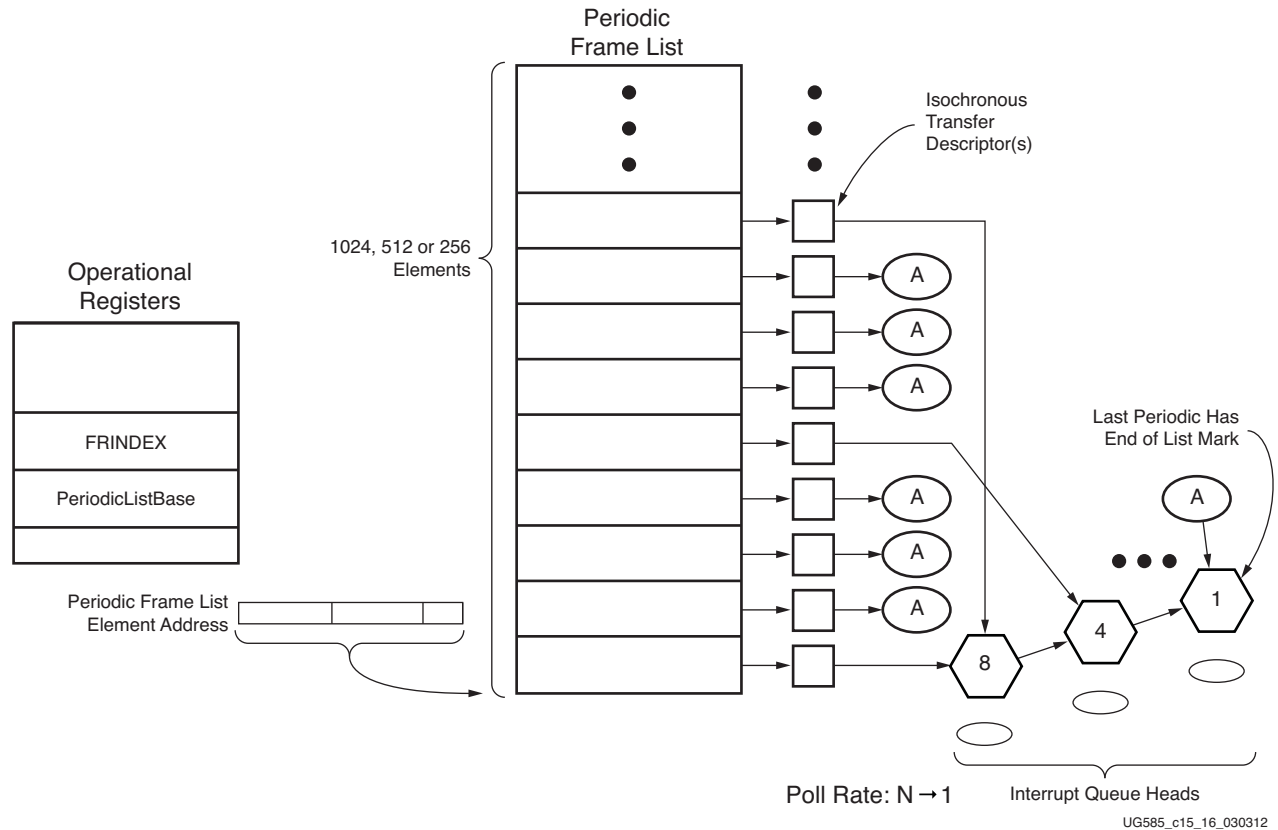


Figure 15-16: Periodic Schedule Organization

The periodic frame list is a 4K-page aligned array of frame list link pointers. The length of the frame list can be programmable. The programmability of the periodic frame list is exported to system software via the HCCPARAMS register. If non-programmable, the length is 1,024 elements. If programmable, the length can be selected by system software as one of 256, 512, or 1,024 elements. An implementation must support all three sizes. Programming the size (i.e., the number of elements) is accomplished by system software writing the appropriate value into frame list size field in the 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 DWORD boundaries within the frame List.

Table 15-16: Format of Frame List

Bits	Description
31:5	Frame List Link Pointer
4	= 0
3	= 0
2:1	TYP
0	T

Frame list link pointers always reference memory objects that are 32-byte aligned. The referenced object might 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 are used to 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 to a one, the host controller will never use 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 being referenced by this pointer. The TYP value encodings are shown in Table 15-17.

Table 15-17: TYP field Encoding

Value	Meaning
00b	Isochronous Transfer Descriptor
01b	Queue Head
10b	Split Transaction Isochronous Transfer Descriptor
11b	Frame Span Traversal Node

15.10.2 Asynchronous List Queue Head Pointer

The asynchronous transfer list (based at the ASYNCLISTADDR register) is where all the control and bulk transfers are managed. Host controllers use this list only when it reaches the end of the periodic list, the periodic list is disabled, or the periodic list is empty.

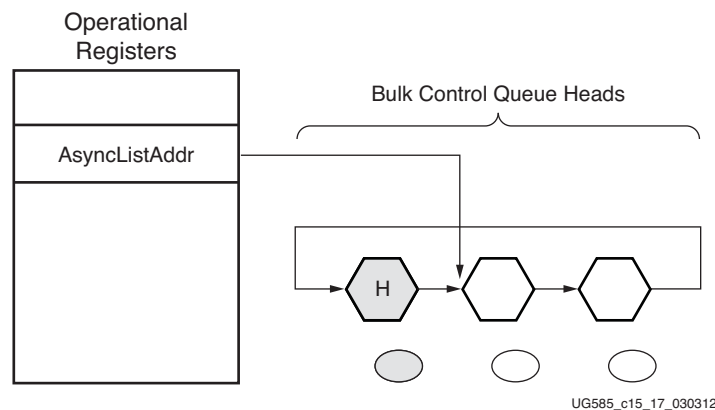


Figure 15-17: Asynchronous Schedule Organization

The asynchronous list is a simple circular list of queue heads. The 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.

15.10.3 Isochronous (High-Speed) Transfer Descriptor (iTID)

The format of an isochronous transfer descriptor is illustrated in Figure 15-18. 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			
Next Link Pointer																												0	0	Typ	T	03-00h		
Status		Transaction 0 Length														IOC	PG*		Transaction 0 Offset*										07-04h					
Status		Transaction 1 Length														IOC	PG*		Transaction 1 Offset*										0B-08h					
Status		Transaction 2 Length														IOC	PG*		Transaction 2 Offset*										0F-0Ch					
Status		Transaction 3 Length														IOC	PG*		Transaction 3 Offset*										13-10h					
Status		Transaction 4 Length														IOC	PG*		Transaction 4 Offset*										17-14h					
Status		Transaction 5 Length														IOC	PG*		Transaction 5 Offset*										1B-18h					
Status		Transaction 6 Length														IOC	PG*		Transaction 6 Offset*										1F-1Ch					
Status		Transaction 7 Length														IOC	PG*		Transaction 7 Offset*										23-20h					
Buffer Pointer (Page 0)																					EndPt		R	Device Address										27-24h
Buffer Pointer (Page 1)																					I/O		Maximum Packet Size										2B-28h	
Buffer Pointer (Page 2)																					Reserved										Mult		2F-2Ch	
Buffer Pointer (Page 3)																					Reserved												33-30h	
Buffer Pointer (Page 4)																					Reserved												37-34h	
Buffer Pointer (Page 5)																					Reserved												3B-38h	
Buffer Pointer (Page 6)																					Reserved												3F-3Ch	



Host Controller Read/Write



Host Controller Read Only

Note: The fields marked with * may be modified by the host controller if the I/O field indicates an OUT.

UG585_c15_18_030312

Figure 15-18: Isochronous Transaction Descriptor (iTID)

15.10.4 Next Link Pointer

The first DWORD of an iTD is a pointer to the next schedule data structure.

Table 15-18: Next Link Pointer

Bits	Description
31:5	Next 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).
2:1	Queue Head Type (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: <ul style="list-style-type: none"> 00b: iTD (isochronous transfer descriptor) 01b: QH (queue head) 10b: siTD (split transaction isochronous transfer descriptor) 11b: FSTN (frame span traversal node)
0	Terminate bit (T) <ul style="list-style-type: none"> 1b: Link Pointer field is not valid. 0b: Link Pointer field is valid.

15.10.5 iTD Transaction Status and Control List

DWORDS 1 through 8 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 X 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 DWORDs of the Buffer Page Pointer list, to execute a transaction on the USB.

Table 15-19: ITD Transaction Status and Control List

Bits	Description
31:28	<p>Status</p> <p>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:</p> <ul style="list-style-type: none"> • 31 Active Set to one 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 zero indicating that a transaction for this element should not be executed when it is next encountered in the schedule. • 30 Data Buffer Error Set to a one 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 (under run). If an overrun condition occurs, no action is necessary. • 29 Babble Detected Set to one by the Host Controller during status update when 'babble' is detected during the transaction generated by this descriptor. • 28 Transaction Error Set to one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit can only be set for isochronous IN transactions.
27:16	<p>Transaction X Length</p> <p>For an OUT, this field is the number of data bytes the host controller will send during the transaction. The host controller is not required to update this field to reflect the actual number of bytes transferred during the transfer.</p> <p>For an IN, the initial value of the field is the number of bytes the host expects the endpoint to deliver. During the status update, the host controller writes back the field the number of bytes successfully received.</p> <p>The value in this register is the actual byte count, e.g.:</p> <ul style="list-style-type: none"> • 0 zero length data • 1 one byte • 2 two bytes, etc. <p>The maximum value this field can contain is 0xC00 (3072).</p>
15	<p>Interrupt On Complete (IOC)</p> <p>If this bit is set to 1, it specifies that when this transaction completes, the Host Controller should issue an interrupt at the next interrupt threshold.</p>
14:12	<p>Page Select (PG)</p> <p>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.</p>
11:0	<p>Transaction X Offset</p> <p>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.</p>

15.10.6 iTD Buffer Page Pointer List (Plus)

DWORDS 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) * 1,024 (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.

Since each pointer is a 4K aligned page pointer, the least significant 12 bits in several of the page pointers are used for other purposes.

Table 15-20: iTD Buffer Pointer Page 0 (Plus)

Bits	Description
31:12	Buffer Pointer (Page 0) This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11:8	Endpoint Number (Endpt) This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Reserved Bit reserved for future use and should be initialized by software to zero.
6:0	Device Address This field selects the specific device serving as the data source or sink.

Table 15-21: iTD Buffer Pointer Page 1 (Plus)

Bits	Description
31:12	Buffer Pointer (Page 1) This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11	Direction (I/O) This field encodes whether the high-speed transaction should use an IN or OUT PID. <ul style="list-style-type: none"> 0b: OUT 1b: 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 (.e.g., per microframe). 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 1,024 (400h). Any value larger yields undefined results.

Table 15-22: iTD Buffer Pointer Page 2 (Plus)

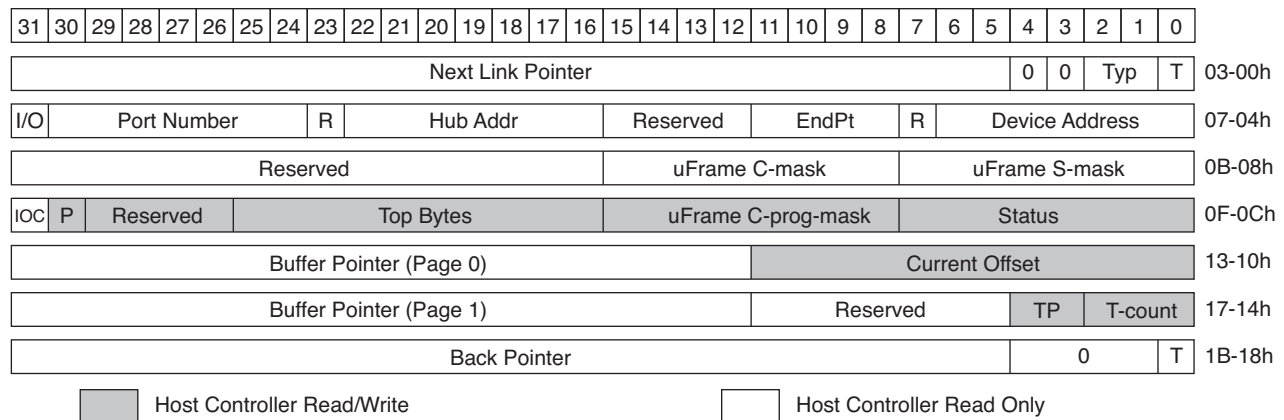
Bits	Description
31:12	Buffer Pointer (Page 2) This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11:2	Reserved This bit reserved for future use and should be set to zero.
1:0	Multi. This field is used to indicate to the host controller the number of transactions that should be executed per transaction description (e.g. per micro-frame). <ul style="list-style-type: none"> 00b: Reserved. A zero in this field yields undefined results 01b: One transaction to be issued for this endpoint per micro-frame 10b: wo transactions to be issued for this endpoint per micro-frame 11b: Three transactions to be issued for this endpoint per micro-frame

Table 15-23: iTD Buffer Pointer Page 3-6

Bits	Description
31:12	Buffer Pointer (Page X) This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11:0	Reserved This bit reserved for future use and should be set to zero.

15.10.7 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.



UG585_c15_19_030312

Figure 15-19: Split-Transaction Isochronous Transaction Descriptor (siTD)

15.10.8 Next Link Pointer

DWORD0 of a siTD is a pointer to the next schedule data structure.

Table 15-24: Next Link Pointer

Bits	Description
31:5	<p>Next Link Pointer</p> <p>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.</p>
2:1	<p>Queue Head Type (Typ)</p> <p>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:</p> <ul style="list-style-type: none"> 00b: iTD (isochronous transfer descriptor) 01b: QH (queue head) 10b: siTD (split transaction isochronous transfer descriptor) 11b: FSTN (frame span traversal node)
0	<p>Terminate bit (T)</p> <ul style="list-style-type: none"> 1b: Link Pointer field is not valid 0b: Link Pointer field is valid

15.10.9 siTD Endpoint Capabilities/Characteristics

DWORDS 1 and 2 specify static information about the full-speed endpoint, the addressing of the parent Companion Controller, and micro-frame scheduling control.

Table 15-25: Endpoint and Transaction Translator Characteristics

Bits	Description
31	Direction (I/O) This field encodes whether the full-speed transaction should use an IN or OUT. <ul style="list-style-type: none"> 1b: IN 0b: OUT
30:24	Port Number This field is the port number of the recipient Transaction Translator.
23	Reserved Bit reserved and should be set to zero.
22:16	Hub Address (Hub Addr) This field holds the device address of the Companion Controllers' hub.
15:12	Reserved Field reserved and should be set to zero.
11:8	Endpoint Number (EndPt) This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Reserved Bit reserved and should be set to zero.
6:0	Device Address This field selects the specific device serving as the data source or sink.

Table 15-26: Micro-Frame Schedule Control

Bits	Description
31:16	Reserved Field reserved and should be set to zero.
15:8	<p>Split Completion Mask (uFrame C-mask)</p> <p>This field (along with the Active and SplitXstate fields in the Status byte) is used to determine during which micro-frames the host controller should execute complete-split transactions. This field is a straight bit position field, so if bit zero is set then the complete-split transaction should occur in the first micro-frame, if bit 1 is set then it should occur in the second micro-frame, and so on. When the criteria for using this field is met, an all zeros value has undefined behavior.</p> <p>The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the uFrame C-Mask field is a one, then this siTD is a candidate for transaction execution.</p> <p>There can be more than one bit in this mask set.</p> <p>The C-Mask can be set for multiple micro frames, as it is not known in which microframe the transaction will complete. So the C-Mask can be set for the micro frame after the S-Mask and all subsequent micro frames thereafter. The C-Mask field should not have a bit set to the same micro-frame as the S-Mask is set to.</p>
7:0	<p>Split Start Mask (uFrame S-mask)</p> <p>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.</p> <p>The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the uFrame S-mask field is a one, then this siTD is a candidate for transaction execution.</p> <p>An all zeros value in this field, in combination with existing periodic frame list has undefined results.</p> <p>This field should have only one bit set to 1 at any given time. Having more than one bit set will result in undefined results.</p>

15.10.10 siTD Transfer State

DWORDS 3 is used to manage the state of the transfer.

Table 15-27: siTD Transfer Status and Control

Bits	Description
31	<p>Interrupt On Complete (IOC)</p> <ul style="list-style-type: none"> 0b: Do not interrupt when transaction is complete 1b: Do interrupt when transaction is complete <p>When the host controller determines that the split transaction has completed it will assert a hardware interrupt at the next interrupt threshold.</p>

30	<p>Page Select (P)</p> <p>Used to indicate which data page pointer should be concatenated with the Current Offset field to construct a data buffer pointer (0 selects Page 0 pointer and 1 selects Page 1). 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).</p>
29:26	<p>Reserved</p> <p>Field reserved and should be set to zero.</p>
25:16	<p>Total Bytes To Transfer (Total Bytes)</p> <p>This field is initialized by software to the total number of bytes expected in this transfer. Maximum value is 1,023 (3FFh).</p>
15:8	<p>uFrame Complete-split Progress Mask (uFrame C-prog-mask)</p> <p>This field is used by the host controller to record which split-completes has been executed.</p>
7:0	<p>Status</p> <p>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:</p> <ul style="list-style-type: none"> • 7 Active Set to one by software to enable the execution of an isochronous split transaction by the Host Controller • 6 ERR Set to a one by the Host Controller when an ERR response is received from the Companion Controller. • 5 Data Buffer Error Set to a one 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 (under run). In the case of an under run, the Host Controller will transmit an incorrect CRC (thus invalidating the data at the endpoint). If an overrun condition occurs, no action is necessary. • 4 Babble Detected Set to 1 by the Host Controller during status update when 'babble' is detected during the transaction generated by this descriptor. • 3 Transaction Error Set to 1 by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit can only be set for isochronous IN transactions. • 2 Missed Micro-Frame The host controller detected that a host-induced holdoff caused the host controller to miss a required complete-split transaction. • 1 Split Transaction State The bit encodings are: <ul style="list-style-type: none"> • 0b: 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. • 1b: 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. • 0 Reserved Bit reserved for future use and should be set to zero.

15.10.11 siTD Buffer Pointer List (plus)

DWORDS 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 DWORD in this section are the 4K (page) aligned buffer pointers. The least significant 12 bits of each DWORD are used as additional transfer state.

Table 15-28: Buffer Page Pointer List (plus)

Bits	Description
31:12	<p>Buffer Pointer List.</p> <p>Bits [31:12] of DWORDs 4 and 5 are 4K paged aligned, physical memory addresses. These bits correspond to physical address bits [31:12] respectively. The lower 12 bits in each pointer are defined and used as specified below. The field P specifies the current active pointer.</p>
11:0	<p><u>Page0</u></p> <ul style="list-style-type: none"> Current Offset. <p>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).</p> <p><u>Page1</u></p> <ul style="list-style-type: none"> Transition position (TP) <p>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:</p> <ul style="list-style-type: none"> 00b: All. The entire full-speed transaction data payload is in this transaction (i.e. less than or equal to 188 bytes). 01b: Begin. This is the first data payload for a full-speed that is greater than 188 bytes. 10b: Mid. This is the middle payload for a full-speed OUT transaction that is larger than 188 bytes. 11b: End. This is the last payload for a full-speed OUT transaction that was larger than 188 bytes. <p>Transaction count (T-Count). Software initializes this field with the number of OUT start-splits this transfer requires. Any value larger than 6 is undefined.</p>

15.10.12 siTD Back Link Pointer

DWORD 6 of a siTD is simply another schedule link pointer. This pointer is always zero, or references a siTD. This pointer cannot reference any other schedule data structure.

Table 15-29: siTD Back Link Pointer

Bits	Description
31:5	siTD Back Pointer (Back Pointer). This field is a physical memory pointer to a siTD.
4:1	Reserved Field reserved and should be set to zero.
0	Terminate bit (T). <ul style="list-style-type: none"> 1b: siTD Back Pointer field is not valid 0b: siTD Back Pointer field is valid

15.10.13 Queue Element Transfer Descriptor (qTD)

This data structure is only used with a queue head. This data structure is used for one or more USB transactions. This data structure is used to transfer up to 20,480 (5*4,096) bytes. The structure contains two structure pointers used for queue advancement, a DWORD 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 can 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.

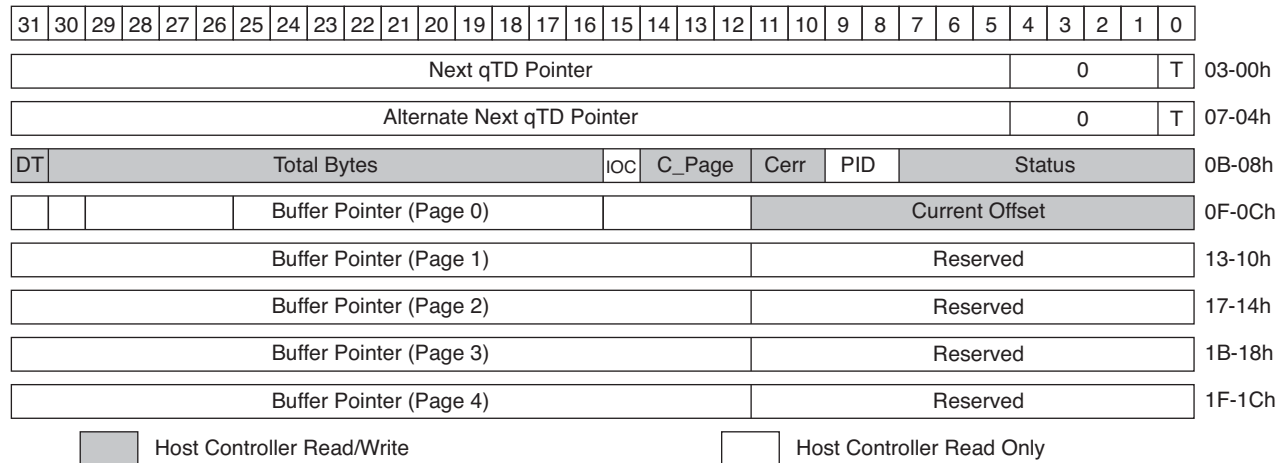


Figure 15-20: Queue Element Transfer Descriptor Block Diagram

Queue Element Transfer Descriptors must be aligned on 32-byte boundaries.

15.10.14 Next qTD Pointer

The first DWORD of an element transfer descriptor is a pointer to another transfer element descriptor.

Table 15-30: qTD Next Element Transfer Pointer (DWORD 0)

Bits	Description
31:5	Next Transfer Element Pointer (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 Field reserved and should be set to zero.
0	Terminate bit (T). <ul style="list-style-type: none"> 1b: Pointer is not valid 0b: Pointer is valid (points to a valid Transfer Element) This bit indicates to the Host controller that there are no more valid entries in the queue.

15.10.15 Alternate Next qTD Pointer

The second DWORD of a queue element transfer descriptor is used to support hardware-only advance of the data stream to the next client buffer on short packet. To be more explicit the host controller will always use this pointer when the current qTD is retired due to short packet.

Table 15-31: qTD Alternate Next Element Transfer Pointer (DWORD 1)

Bits	Description
31:5	Alternate Next Transfer Element Pointer (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 Field reserved and should be set to zero.
0	Terminate bit (T) <ul style="list-style-type: none"> 1b: Pointer is not valid 0b: Pointer is valid (points to a valid Transfer Element) This bit indicates to the Host Controller that there are no more valid entries in the queue.

15.10.16 qTD Token

The third DWORD 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).

Note: The field descriptions forward reference fields defined in the queue head. Where necessary, these forward references are preceded with a QH notation.

Table 15-32: qTD Token (DWORD 2)

Bits	Description
31	Data Toggle (DT) 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) 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, only on the successful completion of the transaction. The maximum value software stored in this field is 5 * 4K (5000H). This is the maximum number of bytes 5 page pointers can access. If the value of this field is zero 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 QHD.Maximum Packet Length. If software builds such a transfer descriptor for an OUT transfer, the last transaction will always be less than QHD.Maximum Packet Length. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K. Therefore, the maximum recommended transfer is 16K (4000H).
15	Interrupt On Complete (IOC) If this bit is set to a one, it specifies that when this qTD is completed, the Host Controller should issue an interrupt at the next interrupt threshold.
14:12	Current Page (C_Page) This field is used as an index into the qTD buffer pointer list. Valid values are in the range 0H to 4H. The host controller is not required to write this field back when the qTD is retired.

11:10	<p>Error Counter (Cerr)</p> <p>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.</p> <p>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 will be generated if the USB Error Interrupt Enable bit in the USBINTR register is set to a one.</p> <p>If HCD programs this field to zero during set-up, the Host controller will not count errors for this qTD and there will be no limit on the retries of this qTD. Note that write-backs of intermediate execution state are to the queue head overlay area, not the qTD.</p> <table data-bbox="386 598 735 741"> <tr> <td>Transaction Error</td><td>Yes</td></tr> <tr> <td>Data Buffer Error</td><td>No⁽³⁾</td></tr> <tr> <td>Stall</td><td>No⁽¹⁾</td></tr> <tr> <td>Babble Detected</td><td>No⁽¹⁾</td></tr> <tr> <td>No Error</td><td>No⁽²⁾</td></tr> </table> <p>Notes:</p> <ol style="list-style-type: none"> 1. Detection of Babble or Stall automatically halts the queue head. Thus, count is not decremented 2. If the EPS field indicates a HS device or the queue head is in the Asynchronous Schedule (and PIDCode indicates an IN or OUT) and a bus transaction completes and the host controller does not detect a transaction error, then 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 (3) 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 00b. 3. Data buffer errors are host problems. They don't count against the device's retries. <p>Note: 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.</p>	Transaction Error	Yes	Data Buffer Error	No ⁽³⁾	Stall	No ⁽¹⁾	Babble Detected	No ⁽¹⁾	No Error	No ⁽²⁾
Transaction Error	Yes										
Data Buffer Error	No ⁽³⁾										
Stall	No ⁽¹⁾										
Babble Detected	No ⁽¹⁾										
No Error	No ⁽²⁾										
9:8	<p>PID Code (PID)</p> <p>This field is an encoding of the token, which should be used for transactions associated with this transfer descriptor. Encodings are:</p> <ul style="list-style-type: none"> • 00b: DUT. Token generates token (E1H) • 01b: IN. Token generates token (69H) • 10b: SETUP. Token generates token (2DH) (undefined if end-point is an Interrupt transfer type, e.g. µFrame S-mask field in the queue head is non-zero.) • 11b: Reserved. 										

7:0	<p data-bbox="358 201 427 226">Status</p> <p data-bbox="358 233 1271 321">his field is used by the Host Controller to communicate individual command execution states back to HCD. This field contains the status of the last transaction performed on this qTD. The bit encodings are:</p> <ul data-bbox="358 327 1289 993" style="list-style-type: none"> <li data-bbox="358 327 1289 420">• 7 Active Set to one by software to enable the execution of transactions by the Host Controller. <li data-bbox="358 426 1289 604">• 6 Halted Set to a one 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 that a transaction results in the Halted bit being set to a one, the Active bit is also set to zero. <li data-bbox="358 611 1289 816">• 5 Data Buffer Error Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overflow) or is unable to supply data fast enough during transmission (under run). If an overflow condition occurs, the Host Controller will force a timeout condition on the USB, invalidating the transaction at the source. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer. <li data-bbox="358 823 1289 993">• 4 Babble Detected Set to a one 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. Since "babble" is considered a fatal error for the transfer, setting the Halted bit to a one insures that no more transactions occur because of this descriptor.
-----	--

7:0 (cont'd)	<ul style="list-style-type: none"> • 3 Transaction Error Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). If the host controller sets this bit to a one, then it remains a one for the duration of the transfer. • 2 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 detected that 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, then it remains a one for the duration of the transfer. • 1 Split Transaction State This bit is ignored by the host controller unless the QH.EPS field indicates a full- or low-speed endpoint. When 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: <ul style="list-style-type: none"> • 0b: Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint. • 1b: Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint. • 0 Ping State/ERR If the QH.EPS field indicates a High-speed device and the PID indicates an OUT endpoint, then this is the state bit for the Ping protocol. The bit encodings are: <ul style="list-style-type: none"> • 0b Do OUT. This value directs the host controller to issue an OUT PID to the endpoint. • 1b Do Ping. This value directs the host controller to issue a PING PID to the endpoint. <p>If the QH.EPS field does not indicate a High-speed device, then this field is used as an error indicator bit. It is set to a one by the host controller whenever a periodic split-transaction receives an ERR handshake.</p>
-----------------	---

15.10.17 qTD Buffer page Pointer List

The last five DWORDs 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 15-33: qTD Buffer Pointer(s) (DWORDs 3-7)

Bits	Description
31:12	<p>Buffer Pointer List (Buffer Pointer)</p> <p>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.</p>
11:0	<p>Current Offset (Reserved)</p> <p>This field is reserved in all pointers except the first one (e.g. 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.</p>

15.10.18 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	
Queue Head Horizontal Link Pointer																												0	Typ	T	03-00h	
RL				C	Maximum Packet Length										H	dtc	EP	EndPt			I	Device Address									07-04h	
Mult		Port Number*							Hub Addr*							uFrame C-mask*							uFrame S-mask*							0B-08h		
Current qTD Pointer																												0		0F-0Ch		
Next qTD Pointer																												0		T	13-10h	
Alternate Next qTD Pointer																												NakCnt		T	17-14h	
DT	Total Bytes															IOC	C_Page		Cerr	PID	Status										1B-18h	
Buffer Pointer (Page 0)																	Current Offset											1F-1Ch				
Buffer Pointer (Page 1)																	Reserved											23-20h				
Buffer Pointer (Page 2)																	Reserved											27-24h				
Buffer Pointer (Page 3)																	Reserved											2B-28h				
Buffer Pointer (Page 4)																	Reserved											2F-2Ch				

* These fields are used exclusively to support split transactions to USB 2.0 Hubs.

Transfer Overlay	<input type="checkbox"/>	Host Controller Read/Write	Transfer Results
Static Endpoint State	<input type="checkbox"/>	Host Controller Read Only	

UG585_c15_21_030312

Figure 15-21: Queue Head Structure Layout

15.10.19 Queue Head Horizontal Link Pointer

The first DWORD 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 can reference a queue head or one of the isochronous transfer descriptors. It must not reference a queue element transfer descriptor.

Table 15-34: Queue Head DWORD 0

Bits	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 Field reserved and should be set to zero.
2:1	Queue Head Type (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: <ul style="list-style-type: none"> • 00b: iTD (isochronous transfer descriptor) • 01b: QH (queue head) • 10b: siTD (split transaction isochronous transfer descriptor) • 11b: FSTN (frame span traversal node)
0	Terminate bit (T) <ul style="list-style-type: none"> • 1b: Pointer field is not valid (last QH) • 0b: Pointer is valid <p>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. This bit is ignored by the host controller 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.</p>

15.10.20 Endpoint Capabilities and Characteristics

The second and third DWORDs of a Queue Head specifies 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 is used to manage full- and low-speed data streams for bulk, control, and interrupt via split transactions to USB2.0 Hub Transaction Translator. There are additional fields used for addressing the hub and scheduling the protocol transactions (for periodic).

The host controller must not modify the bits in this region.

Table 15-35: Endpoint Characteristics: Queue Head DWORD 1

Bits	Description
31:28	NAK Count Reload (NL). This field contains a value, which is used by the host controller to reload the NAK Counter field.
27	Control Endpoint Flag (C). If the QH.EPS field indicates the endpoint is not a high speed device, and the endpoint is a control endpoint, then 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 can contain is 0x400 (1,024).
15	Head of Reclamation List Flag (H). This bit is set by System Software to mark a queue head as being the head of the reclamation list.
14	Data Toggle Control (DTC) This bit specifies where the host controller should get the initial data toggle on an overlay transition. <ul style="list-style-type: none"> 0b: PIgnore DT bit from incoming qTD. Host controller preserves DT bit in the queue head. 1b: 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	Endpoint Speed (EP) This is the speed of the associated endpoint. Bit combinations are: <ul style="list-style-type: none"> 00b: Full-Speed (12 Mb/s) 01b: Low-Speed (1.5 Mb/s) 10b: High-Speed (480 Mb/s) 11b: Reserved This field must not be modified by the host controller.
11:8	Endpoint Number (Endpt) This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Inactivate on Next Transaction (I) This bit is used by system software to request that 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 15-36: Endpoint Capabilities: Queue Head DWORD 2

Bits	Description
31:30	<p>High-Bandwidth Pipe Multiplier (Mult)</p> <p>This field is a multiplier used to key the host controller as the number of successive packets the host controller can 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). The valid values are:</p> <ul style="list-style-type: none"> • 00b: Reserved. A zero in this field yields undefined results. • 01b: One transaction to be issued for this endpoint per micro-frame. • 10b: Two transactions to be issued for this endpoint per micro-frame. • 11b: Three transactions to be issued for this endpoint per micro-frame.
29:23	<p>Port Number</p> <p>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.</p>
22:16	<p>Hub Addr</p> <p>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</p>
15:8	<p>Split Completion Mask (μFrame C-Mask)</p> <p>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. This field is a straight bit position field, so if bit zero is set then the complete-split transaction should occur in the first micro-frame, if bit 1 is set then it should occur in the second micro-frame, and so on.</p> <p>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 FRINDEX register. If the FRINDEX register bits decode to a position where the uFrame C-Mask field is a one, then this queue head is a candidate for transaction execution. There can be more than one bit in this mask set.</p> <p>The C-Mask can be set for multiple micro frames, as it is not known in which microframe the transaction will complete. So the C-Mask can be set for the micro frame after the S-Mask and all subsequent micro frames thereafter. The C-Mask field should not have a bit set to the same micro-frame as the S-Mask is set to.</p>

7:0	<p>Interrupt Schedule Mask (μFrame S-mask)</p> <p>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.</p> <p>The host controller uses the value of the three low-order bits of the FRINDEX register as an index into a bit position in this bit vector. If the uFrame S-mask field has a one at the indexed bit position then this queue head is a candidate for transaction execution.</p> <p>If the EPS field indicates the endpoint is a high-speed endpoint, then the transaction executed is determined by the PID field contained in the execution area.</p> <p>This field is also used to support split transaction types: 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.</p> <p>A zero value in this field, in combination with existing in the periodic frame list has undefined results. This field should have only one bit set to 1 at any given time. Having more than one bit set will result in undefined results.</p>
-----	--

15.10.21 Transfer Overlay

The nine DWORDs 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, then it follows the Queue Head Horizontal Link Pointer to the next queue head. The host controller will never follow 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 DWORD3 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 15-37: Current qTD Link Pointer

Bits	Description
31:5	<p>Current Element Transaction Descriptor Link Pointer</p> <p>This field contains the address Of the current transaction being processed in this queue and corresponds to memory address signals [31:5], respectively.</p>
4:0	<p>Reserved</p> <p>Field reserved and should be set to zero.</p>

The DWORDs 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.

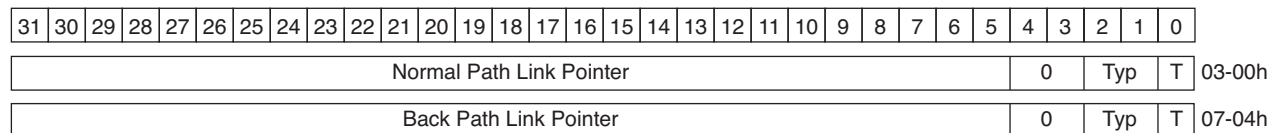
This area is characterized as an overlay because when the queue is advanced to the next queue element, the source queue element is merged onto this area. This area serves an execution cache for the transfer.

Table 15-38: Host-Controller Rules for Bits in Overlay (DWORDs 5, 6, 8 and 9)

DWORD	Bits	Description
5	4:1	<p>NAK Counter (NakCnt)</p> <p>This field is a counter the host controller decrements whenever a transaction for the endpoint associated with this queue head results in a Nak or Nyet response.</p> <p>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.</p>
6	31	<p>Data toggle (DT)</p> <p>The Data Toggle Control controls whether the host controller preserves this bit when an overlay operation is performed.</p>
6	15	<p>Interrupt On Complete (IOC)</p> <p>The IOC control bit is always inherited from the source qTD when the overlay operation is performed.</p>
6	11:10	<p>Error Counter (Cerr)</p> <p>This two-bit field is copied from the qTD during the overlay and written back during queue advancement.</p>
6	0	<p>Ping State (P)/ERR</p> <p>If the EPS field indicates a high-speed endpoint, then this field should be preserved during the overlay operation.</p>
8	7:0	<p>Split-transaction Complete-split Progress (C-prog-mask)</p> <p>This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.</p>
9	11:5	<p>S-bytes</p> <p>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.</p>
9	4:0	<p>Split-transaction Frame Tag (Frame Tag)</p> <p>This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.</p>

15.10.22 Periodic Frame Span Traversal Node (FSTN)

This data structure is to be used 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.



UG585_c15_22_030312

Figure 15-22: Frame Span Traversal Node Structure Layout

15.10.23 STN Normal Path Pointer

The first DWORD of an FSTN contains a link pointer to the next schedule object. This object can be of any valid periodic schedule data type.

Table 15-39: STN Normal Path Pointer

Bits	Description
31:5	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 Field reserved and should be set to zero.
2:1	Queue Head Type (Typ) This field indicates to the Host Controller whether the item referenced is a 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. Value encodings are: <ul style="list-style-type: none"> 00b: iTD (isochronous transfer descriptor) 01b: QH (queue head) 10b: siTD (split transaction isochronous transfer descriptor) 11b: FSTN (frame span traversal node)
0	Terminate bit (T) <ul style="list-style-type: none"> 1b: Link Pointer field is not valid 0b: Link Pointer is valid

15.10.24 FSTN Back Path Link Pointer

The second DWORD 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 to a one, then this FSTN is the Restore indicator. When the T-bit is a one, the host controller ignores the Typ field.

Table 15-40: FSTN Back Path Link Pointer

Bits	Description
31:5	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 Field reserved and should be set to zero.
2:1	Queue Head Type (Typ) Software must ensure this field is set to indicate the target data structure is a Queue Head. Any other value in this field yields undefined results.
0	Terminate bit (T) <ul style="list-style-type: none"> 1b: Link Pointer field is not valid (i.e. 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. 0b: Link Pointer is valid (i.e. the host controller can 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.

15.11 Device Data Structures

This section defines the interface data structures used to communicate control, status, and data between Device Controller Driver (DCD) Software and the Device Controller. The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of device Queue Heads and Transfer Descriptors.

Software must ensure that no interface data structure reachable by the Device Controller spans a 4K-page boundary.

The data structures defined in the chapter 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.

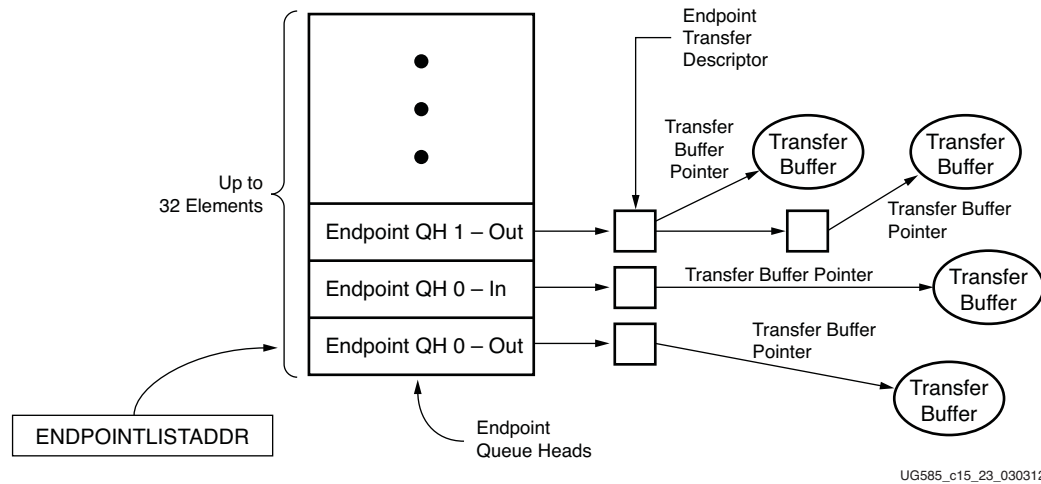


Figure 15-23: End Point Queue Head Organization

Device queue heads are arranged in an array in a continuous area of memory pointed to by the ENDPOINTLISTADDR pointer. The even-numbered device queue heads in the list support receive endpoints (OUT/SETUP) and the odd-numbered queue heads in the list are used for transmit endpoints (IN/INTERRUPT). The device controller will index into this array based upon the endpoint number received from the USB bus. All information necessary to respond to transactions for all primed transfers is contained in this list so the Device Controller can readily respond to incoming requests without having to traverse a linked list.

The Endpoint Queue Head List must be aligned to a 2k boundary.

15.11.1 Endpoint Queue Head (qQH)

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 nextTD pointer DWORD and continues through the end of the buffer pointers DWORDs. After a transfer is complete, the dTD status DWORD is updated in the dTD pointed to by the currentTD pointer. While a packet is in progress, the overlay area of the dQH is used as a staging area for the dTD so that the Device Controller can access needed information with little minimal latency.

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
Mult		zit	0	Maximum Packet Length										ios	0														03-00h		
Current dTD Pointer																									0		07-04h				
Next dTD Pointer																									0		T	0B-08h			
0	Total Bytes										ioc	C_Page	MultO	0	Status										0F-0Ch						
Buffer Pointer (Page 0)												Current Offset												13-10h							
Buffer Pointer (Page 1)												Reserved												17-14h							
Buffer Pointer (Page 2)												Reserved												1B-18h							
Buffer Pointer (Page 3)												Reserved												1F-1Ch							
Buffer Pointer (Page 4)												Reserved												23-20h							
Reserved																											27-24h				
Setup Buffer Bytes 3..0																											2B-28h				
Setup Buffer Bytes 7..4																											2F-2Ch				



Host Controller Read/Write



Host Controller Read Only

Transfer Results

UG585_c15_24_030312

15.11.2 Endpoint Capabilities/Characteristics

Table 15-41: Endpoint Capabilities/Characteristics

Bits	Description
31:30	<p>High-Bandwidth Pipe Multiplier (Mult). This field is used to indicate the number of packets executed per transaction description as given by the following:</p> <ul style="list-style-type: none"> • 00b: 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) • 01b: Execute 1 Transaction • 10b: Execute 2 Transactions • 11b: Execute 3 Transactions <p>Non-ISO endpoints must set Mult = 00b. ISO endpoints must set Mult = 01b, 10b, or 11b as needed.</p>
29	<p>Zero Length Termination Select his bit is used to indicate when a zero length packet is used to terminate transfers where to total transfer length is a multiple. This bit is not relevant for Isochronous.</p> <ul style="list-style-type: none"> • 0b: Enable zero length packet to terminate transfers equal to a multiple of the Maximum Packet Length. • 1b: Disable the zero length packet on transfers that are equal in length to a multiple Maximum Packet Length.

28:27	Reserved Field reserved and should be set to zero.
26:16	Maximum Packet Length This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field can contain is 0x400 (1,024).
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 Field reserved and should be set to zero.

15.11.3 Transfer Overlay

The seven DWORDs 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, then it will not read the associated endpoint.

After an endpoint is readied, the dTD will be 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 will write the results back to the original transfer descriptor and advance the queue.

See section [15.11.6 Endpoint Transfer Descriptor \(qTD\)](#) for a description of the overlay fields.

15.11.4 Current qTD Pointer

The current dTD pointer is used by the device controller to locate the transfer in progress. This word is for Device Controller (hardware) use only and should not be modified by DCD software.

Table 15-42: Next dTD Pointer

Bits	Description
31:5	Current dTD This field is a pointer to the dTD that is represented in the transfer overlay area. This field will be modified by the Device Controller to next dTD pointer during endpoint priming or queue advance.
4:0	Reserved Field reserved and should be set to zero.

15.11.5 Setup Buffer

The setup buffer is dedicated storage for the 8-byte data that follows a set-up PID.

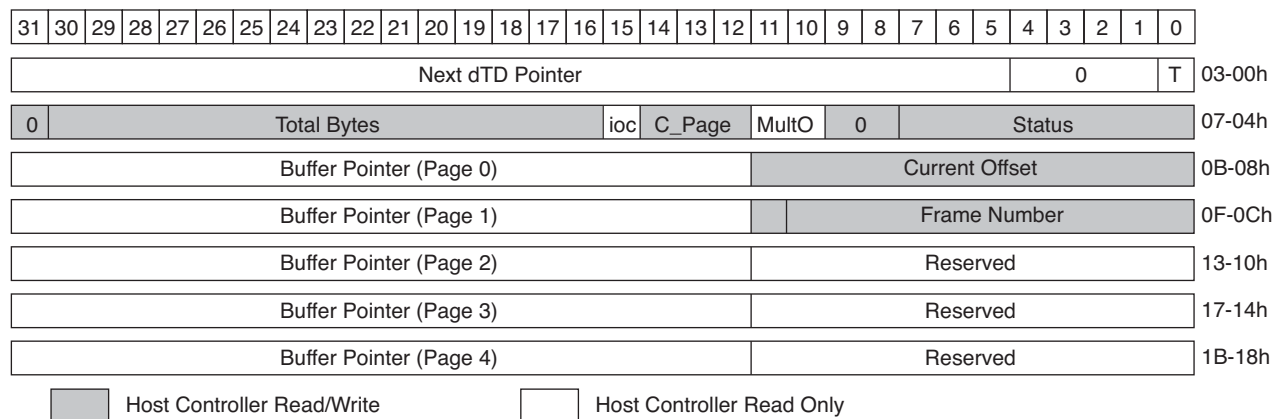
Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is used for receiving setup data packets

Table 15-43: Setup Buffer Bytes

DWORD	Bits	Description
1	31:0	Setup Buffer 0 This buffer contains bytes 3 to 0 of an incoming setup buffer packet and is written by the device controller to be read by software.
2	31:0	Setup Buffer 1 This buffer contains bytes 7 to 4 of an incoming setup buffer packet and is written by the device controller to be read by software.

15.11.6 Endpoint Transfer Descriptor (qTD)

The dTD describes to the device controller the location and quantity of data to be sent/received for given transfer. The DCD should not attempt to modify any field in an active dTD except the Next Like Pointer, which should only be modified as described in the *Reference Manual*.



UG585_c15_25_030312

Figure 15-25: Endpoint Transfer Descriptor (dTD)

Table 15-44: Next dTD Pointer

Bits	Description
31:5	Next Transfer Element Pointer (Next dTD 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 Field reserved and should be set to zero.

0	<p>Terminate (T)</p> <ul style="list-style-type: none"> • 1b: Pointer is not valid • 0b: Pointer is valid (points to a valid Transfer Element Descriptor) <p>This bit indicates to the Device Controller that there are no more valid entries in the queue.</p>
---	---

Table 15-45: dTD Token

Bits	Description
31	<p>Reserved</p> <p>Field reserved and should be set to zero</p>
30:16	<p>Total Bytes</p> <p>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.</p> <p>The maximum value software stored in the field is 5*4K (5000H). This is the maximum number of bytes 5 page pointers can access. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K. Therefore, the maximum recommended transfer is 16K (4000H).</p> <p>If the value of the field is zero 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.</p> <p>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 will always be less than Maximum Packet Length.</p>
15	<p>Interrupt On Complete (IOC)</p> <p>This bit is used to indicate if USBINT is to be set in response to device controller being finished with this dTD.</p>
14:12	<p>Current Page (C_Page)</p> <p>This has the same behavior as in host mode but has no meaning for the SW device driver. Should be ignored.</p>

11:10	<p>Multiplier Override (MultO)</p> <p>This field can be used for transmit ISOs (e.g., ISO-IN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit ISO.</p> <p><u>Examples:</u></p> <p>If:</p> <p style="padding-left: 40px;">QH.multiplier = 3 Maximum packet size = 8 Total Bytes = 15 MultiO = 0 [default]</p> <p>Three packets are sent: {Data2(8); Data1(7); Data0(0)}</p> <p>If:</p> <p style="padding-left: 40px;">QH.multiplier = 3 Maximum packet size = 8 Total Bytes = 15 MultiO = 2</p> <p>Two packets are sent: {Data1(8); Data0(7)}</p> <p>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.</p>
9:8	<p>Reserved</p> <p>Field reserved and should be set to zero.</p>
7:0	<p>Status</p> <p>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:</p> <ul style="list-style-type: none"> • 7 Active • 6 Halted • 5 Data Buffer Error • 3 Transaction Error • 4,2,0, Reserved

Table 15-46: dTD Buffer Page Pointer List

Bits	Description
31:12	<p>Buffer Pointer.</p> <p>Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers</p>
11:0	<p>Current Offset.</p> <p>Offset into the 4 Kb buffer where the packet is to begin.</p>
10:0	<p>Frame Number</p> <p>Written by the device controller to indicate the frame number in which a packet finishes. This is typically be used to correlate relative completion times of packets on an ISO endpoint.</p>

See the controller section in [Appendix B, Register Details](#), for details of register fields.

15.12 I/O Signals

The ULPI, port indicator and power signals are shown in [Table 15-47](#).

The 7z010 CLG225 device supports 32 MIO pins. Pin restrictions are shown in the MIO table in [section 2.4.4 MIO-at-a-Glance Table](#).

Table 15-47: USB ULPI, Port Indicator and Power Signals

USB Port Signals	MIO Pins				EMIO Signals		Default Input Value to Controller
	USB 0	USB 1	I/O	Name	Name	I/O	
Transmit and receive data 4	28	40	IO	USB{0,1}_ULPI_DATA4	~	~	~
Data bus direction control	29	41	I	USB{0,1}_ULPI_DIR	~	~	0
Stop the Transfer (end/interrupt)	30	42	O	USB{0,1}_ULPI_STP	~	~	~
Data flow control signal	31	43	I	USB{0,1}_ULPI_NXT	~	~	0
Transmit and receive data 0	32	44	IO	USB{0,1}_ULPI_DATA0	~	~	~
Transmit and receive data 1	33	45	IO	USB{0,1}_ULPI_DATA1	~	~	~
Transmit and receive data 2	34	46	IO	USB{0,1}_ULPI_DATA2	~	~	~
Transmit and receive data 3	35	47	IO	USB{0,1}_ULPI_DATA3	~	~	~
Transceiver clock for ULPI	36	48	I	USB{0,1}_ULPI_CLK	~	~	
Transmit and receive data 5	37	49	IO	USB{0,1}_ULPI_DATA5	~	~	~
Transmit and receive data 6	38	50	IO	USB{0,1}_ULPI_DATA6	~	~	~
Transmit and receive data 7	39	51	IO	USB{0,1}_ULPI_DATA7	~	~	~
Port Indicator	~	~	~	~	EMIOUSB{0,1}PORTINDCTL{0,1}	O	~
Power Fault	~	~	~	~	EMIOUSB{0,1}VBUSPWRFAULT	I	
Power Select	~	~	~	~	EMIOUSB{0,1}VBUSPWRSELECT	O	~

Gigabit Ethernet Controller

16.1 Introduction

The Gigabit Ethernet Controller (GEM) implements a 10/100/1000 Mb/s Ethernet MAC compatible with the IEEE 802.3-2008 standard capable of operating in either half or full duplex mode at all three speeds. The PS is equipped with two Gigabit Ethernet Controllers. Each controller can be configured independently. The Reduced Gigabit Media Independent Interface (RGMII) of each controllers has access to pins via the MIO and GMII interface access to PL signals via the EMIO interface.

Other Ethernet communications interfaces can be created in the PL using the GMII available on the EMIO interface. For example, the PL can be used to implement these interfaces:

- SGMII and 1000 Base-X, in devices with GTX
- RGMII v2.0 for PHY devices with HSTL Class 1 drivers and receivers

Registers are used to configure the features of the MAC, select different modes of operation, and enable and monitor network management statistics. The DMA controller connects to memory through an AHB bus interface. It is attached to the controller's FIFO interface of the MAC to provide a scatter-gather type capability for packet data storage in an embedded processing system.

The controllers provide MDIO interfaces for PHY management. The PHYs can be controlled from either of the MDIO interfaces.

16.1.1 Block Diagram

A block diagram of one Ethernet controller is shown in Figure 16-1.

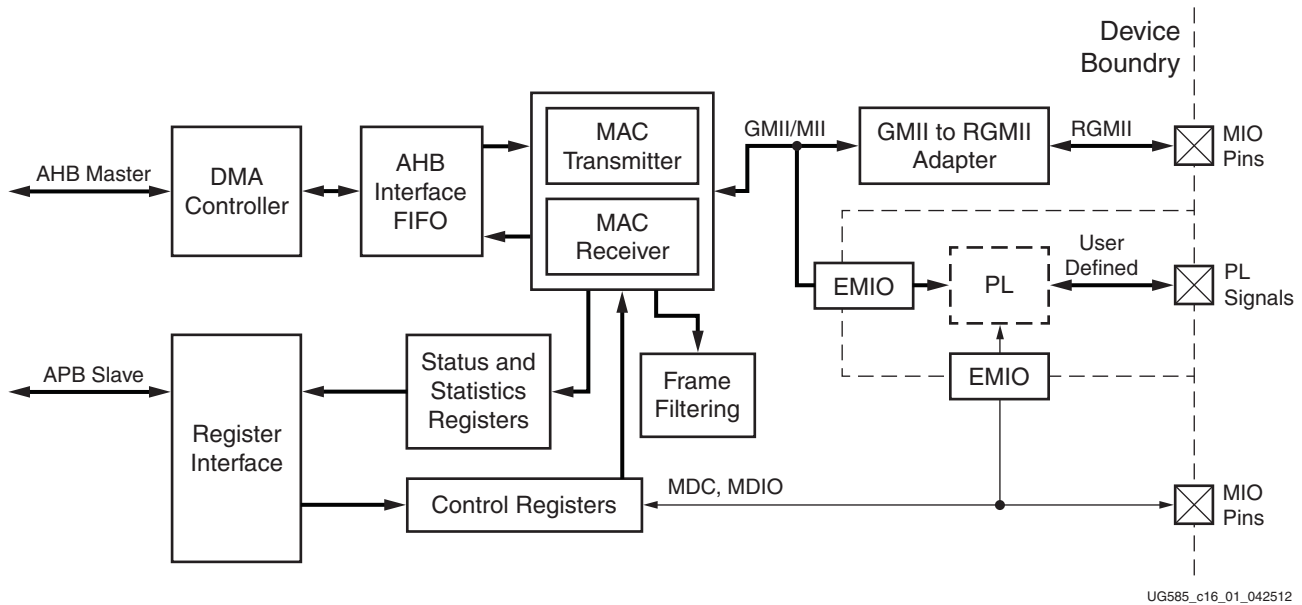


Figure 16-1: Ethernet Controller

16.1.2 Features

Each Gigabit Ethernet MAC controller has the following features:

- IEEE Standard 802.3-2008 compatible, supporting 10/100/1000 Mb/s transfer rates
- Full and half duplex operation at all three speeds of operation
- RGMII interface with external PHY
- GMII/MII interface to the PL to allow connection of interfaces such as TBI, SGMII, 1000 Base-X and RGMII v2.0 support using soft cores (Note: SGMII and 1000 Base-X interfaces require a gigabit transceiver, MGT)
- MDIO interface for physical layer management
- 32-bit AHB DMA master, 32-bit APB bus for control registers access
- Scatter-gather DMA capability
- Interrupt generation to signal receive and transmit completion, or errors and wakeup
- Frame extension and frame bursting at 1,000 Mb/s in half duplex mode
- Automatic pad and cyclic redundancy check (CRC) generation on transmitted frames
- Automatic discard of frames received with errors
- Programmable IPG stretch
- Full duplex flow control with recognition of incoming pause frames and hardware generation of transmitted pause frames

- Address checking logic for four specific 48-bit addresses, four type ID values, promiscuous mode, external address checking, hash matching of unicast and multicast destination addresses and Wake-on-LAN
- 802.1Q VLAN tagging with recognition of incoming VLAN and priority tagged frames
- Supports Ethernet loopback mode
- IPv4 and IPv6 transmit and receive IP, TCP and UDP checksum offload
- Recognition of 1588 rev. 2 PTP frames
- Statistics counter registers for RMON/MIB

16.1.3 System Viewpoint

Figure 16-2 shows Zynq system viewpoint for the Gigabit Ethernet controllers.

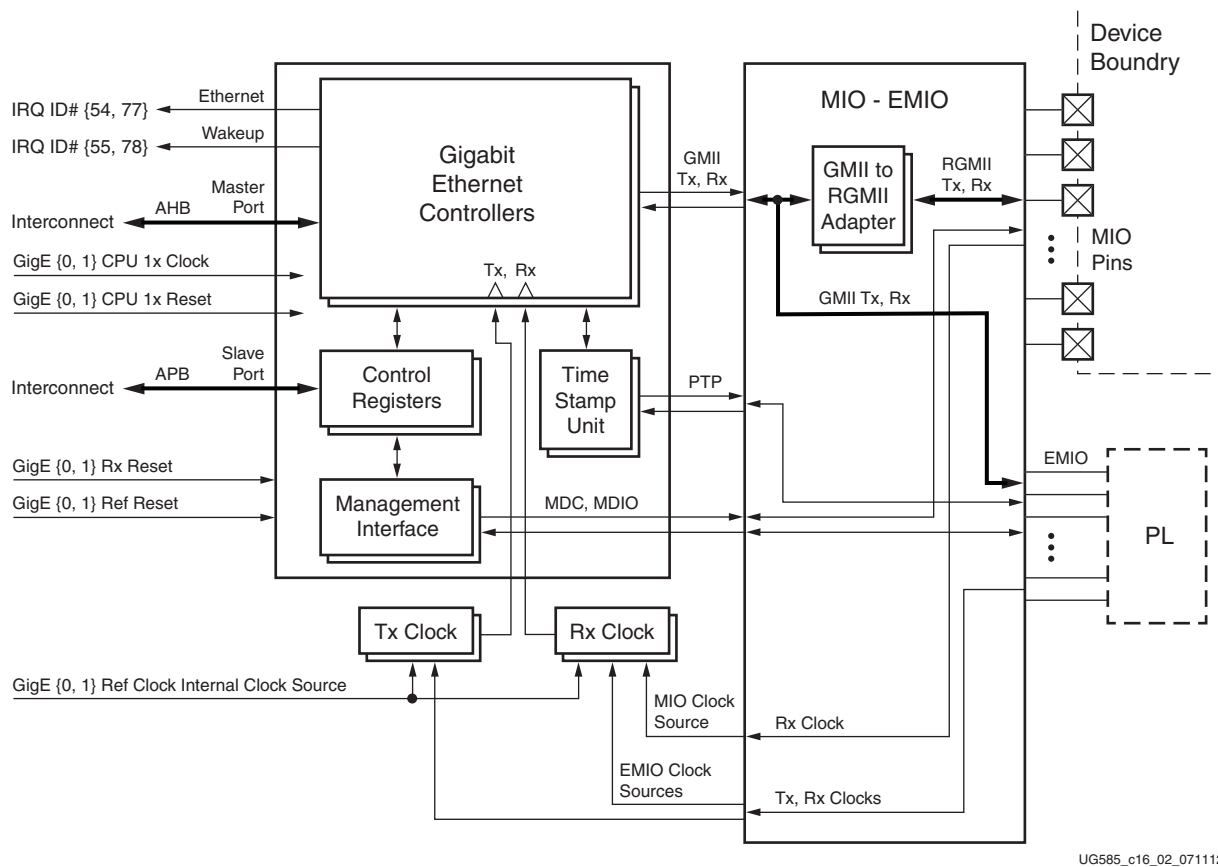


Figure 16-2: System Viewpoint

16.1.4 Notices

7z010 CLG225 Device

This device supports 32 MIO pins and at most one Ethernet interface through the MIO pins. This is shown in the MIO table in section [2.4.4 MIO-at-a-Glance Table](#). One or both of the Ethernet controllers can interface to logic in the PL.

Jumbo Frames

Jumbo frames are not supported.

16.2 Functional Description and Programming Model

The controller comprises four main components:

- MAC controlling transmit, receive, address checking, and loopback
- Control and status registers, statistics registers, and synchronization logic
- DMA controlling DMA transmit, DMA receive and AHB
- Time stamp unit (TSU) for calculating the *IEEE 1588* timer values

10/100/1000 Operation

The gigabit enable bit in the Network Configuration register selects between 10/100 Mb/s Ethernet operation and 1000 Mb/s mode. The 10/100 Mb/s speed bit in the network configuration register is used to select between 10 Mb/s and 100 Mb/s.

MDIO Interface

Both controllers provide MDIO interfaces, however, only one interface is needed to control both of the external PHYs.

16.2.1 MAC Transmitter

The MAC transmitter can operate in either half duplex or full duplex mode, and transmits frames in accordance with the Ethernet IEEE 802.3 standard. In half duplex mode, the CSMA/CD protocol of the IEEE 802.3 specification is followed.

A small input buffer receives data through the AHB interface FIFO and extracts data in either 32-bit form. All subsequent processing prior to the final output is performed in bytes. Transmit data can be output using the GMII/MII interface.

Frame assembly starts by adding the preamble and the start frame delimiter. Data is taken from the transmit FIFO a word at a time. When the controller is configured for gigabit operation, the data output to the PHY uses all eight bits of the txd[7:0] output. In 10/100 mode, transmit data to the PHY is nibble wide and least significant nibble first using txd[3:0] with txd[7:4] tied to logic 0.

If necessary, padding is added to take the frame length to 60 bytes. CRC is calculated using an order 32 bit polynomial. This is inverted and appended to the end of the frame taking the frame length to a minimum of 64 bytes. If the no-CRC bit is set in the second word of the last buffer descriptor of a transmit frame, neither pad nor CRC are appended. The no-CRC bit can also be set through the FIFO.

In full duplex mode (at all data rates), frames are transmitted immediately. Back-to-back frames are transmitted at least 96 bit times apart to guarantee the interframe gap.

In half duplex mode, the transmitter checks carrier sense. If asserted, the transmitter waits for the signal to become inactive, and then starts transmission after the interframe gap of 96 bit times. If the collision signal is asserted during transmission, the transmitter transmits a jam sequence of 32 bits taken from the data register and then retries transmission after the back off time has elapsed. If the collision occurs during either the preamble or SFD, then these fields are completed prior to generation of the jam sequence.

The back off time is based on an XOR of the 10 least significant bits of the data coming from the transmit FIFO and a 10-bit pseudo random number generator. The number of bits used depends on the number of collisions seen. After the first collision 1 bit is used, then the second 2 bits and so on up to the maximum of 10 bits. All 10 bits are used above ten collisions. An error is indicated and no further attempts are made if 16 consecutive attempts cause a collision. This operation is compatible with the description in *Clause 4.2.3.2.5 of the IEEE 802.3 standard* which refers to the truncated binary exponential back off algorithm.

In 10/100 mode, both collisions and late collisions are treated identically, and back off and retry are performed up to 16 times. When operating in gigabit mode, late collisions are treated as an exception and transmission is aborted, without retry. This condition is reported in the transmit buffer descriptor word 1 (late collision, bit 26) and also in the Transmit Status register (late collision, bit 7). An interrupt can also be generated (if enabled) when this exception occurs, and bit 5 in the Interrupt Status Register is set.

When operating in gigabit mode (half duplex) both carrier extension and frame bursting are performed in accordance with the IEEE 802.3 standard. For frames less than 512 bytes carrier extension is used to ensure the minimum slot time is not violated.

Frame bursting is used by the transmitter in gigabit mode (half duplex) when more than one frame is queued for transmission. The first frame of a burst must be carrier extended (if necessary) to ensure that the minimum slot time of 512 bytes is achieved, after which all subsequent frames within the burst must only satisfy the minimum frame length of 64 bytes or greater. Each interframe gap within the burst is filled by the transmitter with carrier extensions, thus ensuring control of the medium is not given up. Several frames may be transmitted up to the burst limit of 65,536 bytes. The transmitter relinquishes control of the medium when there are no more frames queued for transmission or the burst limit is exceeded.

In gigabit mode any collisions occurring after the minimum slot time for the first frame within a burst are treated as a late collision. The burst is terminated upon this event.

When bit [28] is set in the Network Configuration register the IPG can be stretched beyond 96 bits depending on the length of the previously transmitted frame and the value written to the IPG_STRETCH register. The least significant 8 bits of the IPG_STRETCH register multiply the previous frame length (including preamble) the next significant 8 bits (+1 so as not to get a divide by zero) divide the frame length to generate the IPG. IPG stretch only works in full duplex mode and when bit 28 is set in the Network Configuration register. The IPG_STRETCH register cannot be used to shrink the IPG below 96 bits.

If the back pressure bit is set in the Network Control register or if the half_duplex_flow_control_en input is set in 10M or 100M half duplex mode, the transmit block transmits 64 bits of data, which can consist of 16 nibbles of 1011 or in bit rate mode 64 1s, whenever it sees an incoming frame to force a collision. This provides a way of implementing flow control in half duplex mode. Note that this feature is not available in gigabit half duplex mode.

16.2.2 MAC Receiver

All processing within the MAC receiver uses 16-bit data paths. The MAC receiver checks for valid preamble, FCS, alignment, and length. It then sends the received frames to the FIFO (to either the DMA controller or external to the IP core) and stores the frames destination address for use by the address checking block.

When the TBI is selected, 16 bit words are passed directly from the PCS layer to the MAC receiver, except in 10 and 100 Mb/s SGMII modes when 8 bits are read. In 10 and 100 Mb/s SGMII modes the MAC rx_clk runs more slowly than the PCS rxc clock and the same 8 bits of data is read 5 or 50 times.

If, during frame reception, the frame is found to be too long, a bad frame indication is sent to the FIFO. The receiver logic ceases to send data to memory as soon as this condition occurs.

At end of frame reception the receive block indicates to the DMA block whether the frame is good or bad. The DMA block recovers the current receive buffer if the frame was bad.

Ethernet frames are normally stored in DMA memory or to the FIFO complete with the FCS. Setting the FCS remove bit in the network configuration register (bit [17]) causes frames to be stored without their corresponding FCS. The reported frame length field is reduced by four bytes to reflect this operation.

The receive block signals to the register block to increment the alignment, CRC (FCS), short frame, long frame, jabber or receive symbol errors when any of these exception conditions occur.

If bit [26] is set in the network configuration CRC errors are ignored and frames with CRC errors are not discarded, though the Frame Check Sequence Errors Statistic register is still incremented. Bit[13] of the receiver descriptor word 1 is updated to indicate the FCS validity for the particular frame. This is useful for applications where individual frames with FCS errors must be identified.

Received frames can be checked for length field error by setting the length field error frame discard bit of the Network Configuration register (bit [16]). When this bit is set, the receiver compares a frame's measured length with the length field (bytes 13 and 14) extracted from the frame. The frame is discarded if the measured length is shorter. This checking procedure is for received frames between 64 bytes and 1,518 bytes in length.

Each discarded frame is counted in the 10-bit length field Error Statistics register. Frames where the length field is greater than or equal to 0x0600 are not checked.

When operating in gigabit mode (half duplex), the receiver discards frames that do not meet the minimal slot time of 512 bytes. If a burst is detected, the first frame is checked to ensure it meets the slot time, but all subsequent frames of the burst are checked to ensure that they meet the minimum frame size of 64 bytes.

In gigabit mode (half duplex), carrier extension errors are detected by the receiver during the minimum slot time, and the frame discarded. An error of this nature causes the Receive Symbol Errors Statistic register to be incremented. Carrier extension errors occurring during the inter packet gap period are ignored and have no effect on the statistics.

16.2.3 MAC Filtering

The MAC filter determines which frames should be written to the AHB interface FIFO and onto the DMA controller.

Whether a frame is passed depends on what is enabled in the Network Configuration register, the state of the external matching pins, the contents of the specific address, type, and hash registers and the frame's destination address and type field.

If bit [25] of the Network Configuration register is not set, a frame is not copied to memory if the Gigabit Ethernet controller is transmitting in half duplex mode at the time a destination address is received.

Ethernet frames are transmitted a byte at a time, least significant bit first. The first six bytes (48 bits) of an Ethernet frame make up the destination address. The first bit of the destination address, which is the LSB of the first byte of the frame, is the group or individual bit. This is one for multicast addresses and zero for unicast. The all-ones address is the broadcast address and a special case of multicast.

The Gigabit Ethernet controller supports recognition of four specific addresses. Each specific address requires two registers, Specific Address register bottom and Specific Address register top. Specific address register bottom stores the first four bytes of the destination address and Specific Address register top contains the last two bytes. The addresses stored can be specific, group, local or universal.

The destination address of received frames is compared against the data stored in the Specific Address registers once they have been activated. The addresses are deactivated at reset or when their corresponding Specific Address register bottom is written. They are activated when Specific Address register top is written. If a receive frame address matches an active address, the frame is written to the FIFO and on to DMA controller, if used.

Frames can be filtered using the type ID field for matching. Four type ID registers exist in the register address space and each can be enabled for matching by writing a one to the MSB (bit [31]) of the respective register. When a frame is received, the matching is implemented as an OR function of the various types of match.

The contents of each type ID registers (when enabled) are compared against the length/type ID of the frame being received (e.g., bytes 13 and 14 in non-VLAN and non-SNAP encapsulated frames)

and copied to memory if a match is found. The encoded type ID match bits (Word 0, bit [22] and bit [23]) in the receive buffer descriptor status are set indicating which type ID register generated the match, if the receive checksum offload is disabled.

The reset state of the type ID registers is zero, hence each is initially disabled.

The following example illustrates the use of the address and type ID match registers for a MAC address of 21:43:65:87:A9:CB

Preamble	55
SFD	D5
DA (Octet 0 - LSB)	21
DA (Octet 1)	43
DA (Octet 2)	65
DA (Octet 3)	87
DA (Octet 4)	A9
DA (Octet 5 - MSB)	CB
SA (LSB)	00*
SA	00*
SA	00*
SA	00*
SA	00*
SA (MSB)	00*
Type ID (MSB)	43
Type ID (LSB)	21

Note: * – contains the address of the transmitting device.

The sequence above shows the beginning of an Ethernet frame. Byte order of transmission is from top to bottom as shown. For a successful match to specific address 1, the following address matching registers must be set up:

Specific address 1 bottom (Address 0x088)	0x87654321
Specific address 1 top (Address 0x08C)	0x0000CBA9

And for a successful match to the type ID, the following type ID match 1 register must be set up:

Type ID Match 1 (Address 0x0A8)	0x80004321
---------------------------------	------------

Broadcast Address

Frames with the broadcast address of 0xFFFFFFFFFFFFFF are stored to memory only if the 'no broadcast' bit in the Network Configuration register is set to zero.

Hash Addressing

The Hash Address register is 64 bits long and takes up two locations in the memory map. The least significant bits are stored in Hash register bottom and the most significant bits in Hash register top.

The unicast hash enable and the multicast hash enable bits in the Network Configuration register enable the reception of hash matched frames. The destination address is reduced to a 6 bit index into the 64 bit hash register using the following hash function. The hash function is an XOR of every sixth bit of the destination address.

```

hash_index[05] = da[05]^da[11]^da[17]^da[23]^da[29]^da[35]^da[41]^da[47]
hash_index[04] = da[04]^da[10]^da[16]^da[22]^da[28]^da[34]^da[40]^da[46]
hash_index[03] = da[03]^da[09]^da[15]^da[21]^da[27]^da[33]^da[39]^da[45]
hash_index[02] = da[02]^da[08]^da[14]^da[20]^da[26]^da[32]^da[38]^da[44]
hash_index[01] = da[01]^da[07]^da[13]^da[19]^da[25]^da[31]^da[37]^da[43]
hash_index[00] = da[00]^da[06]^da[12]^da[18]^da[24]^da[30]^da[36]^da[42]

```

da[0] represents the least significant bit of the first byte received, that is, the multicast/unicast indicator, and da[47] represents the most significant bit of the last byte received.

If the hash index points to a bit that is set in the Hash register then the frame is matched according to whether the frame is multicast or unicast.

A multicast match is signaled if the multicast hash enable bit is set, da[0] is logic 1 and the hash index points to a bit set in the Hash register. A unicast match is signaled if the unicast hash enable bit is set, da[0] is logic 0 and the hash index points to a bit set in the Hash register. To receive all multicast frames, the Hash register should be set with all ones and the multicast hash enable bit should be set in the Network Configuration register.

Copy All Frames (or Promiscuous Mode)

If the copy all frames bit is set in the Network Configuration register then all frames (except those that are too long, too short, have FCS errors, or have rx_er asserted during reception) are copied to memory. Frames with FCS errors are copied if bit [26] is set in the Network Configuration register.

Disable Copy of Pause Frames

Pause frames can be prevented from being written to memory by setting the disable copying of pause frames control bit [23] in the Network Configuration register. When set, pause frames are not copied to memory regardless of the copy all frames bit, whether a hash match is found, a type ID match is identified, or if a destination address match is found.

VLAN Support

An Ethernet encoded 802.1Q VLAN tag is shown in [Table 16-1](#)

Table 16-1: VLAN Tag Control Information

TPID (Tag Protocol Identifier) 16 Bits	TCI (Tag Control Information) 16 Bits
0x8100	First 3 bits priority, then CFI bit, last 12 bits VID

The VLAN tag is inserted at the 13th byte of the frame adding an extra four bytes to the frame. To support these extra four bytes, the Gigabit Ethernet controller can accept frame lengths up to 1,536 bytes by setting bit [8] in the Network Configuration register.

If the VID (VLAN identifier) is null (0x000) a priority-tagged frame is indicated.

The following bits in the receive buffer descriptor status word provide information about VLAN tagged frames:

- Bit [21] set if receive frame is VLAN tagged (i.e. type id of 0x8100).
- Bit [20] set if receive frame is priority tagged (i.e. type id of 0x8100 and null VID). (If bit [20] is set bit [21] is also set).
- Bits [19], [18] and [17] set to priority if bit [21] is set.
- Bit [16] set to CFI if bit [21] is set.

The controller can be configured to reject all frames except VLAN tagged frames by setting the discard non-VLAN frames bit in the Network Configuration register.

16.2.4 Wake on LAN Support

The receive block supports Wake on LAN by detecting the following events on incoming receive frames:

- Magic packet
- ARP request to the device IP address
- Specific address 1 filter match
- Multicast hash filter match

If one of these events occurs, Wake on LAN detection is indicated by asserting the wakeup interrupt. These events can be individually enabled through bits[19:16] of the Wake on LAN register. Also, for Wake on LAN detection to occur receive enable must be set in the Network Control register, however a receive buffer does not have to be available.

wakeup interrupt assertion due to ARP request, specific address 1, or multicast filter events occur even if the frame is in error. For magic packet events, the frame must be correctly formed and error free.

A magic packet event is detected if all of the following are true:

- Magic packet events are enabled through bit [16] of the Wake on LAN register
- The frame's destination address matches specific address 1
- The frame is correctly formed with no errors
- The frame contains at least 6 bytes of 0xFF for synchronization
- There are 16 repetitions of the contents of Specific Address 1 register immediately following the synchronization

An ARP request event is detected if all of the following are true:

- ARP request events are enabled through bit [17] of the Wake on LAN register
- Broadcasts are allowed by bit 5 in the Network Configuration register
- The frame has a broadcast destination address (bytes 1 to 6)
- The frame has a typeID field of 0x0806 (bytes 13 and 14)

- The frame has an ARP operation field of 0x0001 (bytes 21 and 22)
- The least significant 16 bits of the frame's ARP target protocol address (bytes 41 and 42) match the value programmed in bits[15:0] of the Wake on LAN register

The decoding of the ARP fields adjusts automatically if a VLAN tag is detected within the frame. The reserved value of 0x0000 for the Wake on LAN target address value does not cause an ARP request event, even if matched by the frame.

A specific address 1 filter match event occurs if all of the following are true:

- Specific address 1 events are enabled through bit [18] of the Wake on LAN register
- The frame's destination address matches the value programmed in the Specific Address 1 registers

A multicast filter match event occurs if all of the following are true:

- Multicast hash events are enabled through bit [19] of the Wake on LAN register
- Multicast hash filtering is enabled through bit [6] of the Network Configuration register
- The frame destination address matches against the multicast hash filter
- The frame destination address is not a broadcast

16.2.5 DMA Block

The DMA controller is attached to the FIFO to provide a scatter-gather type capability for packet data storage in an embedded processing system.

Packet Buffer DMA

The controller uses a packet buffer which has the following features

- 32 data bus width support
- Easier to guarantee maximum line rate due to the ability to store multiple frames in the packet buffer
- More efficient use of the AHB interface
- Full store and forward
- Support for Transmit TCP/IP checksum offload
- Support for priority queuing
- When a collision on the line occurs during transmission, the packet is automatically replayed directly from the packet buffer memory rather than having to re-fetch through the AHB interface
- Received error packets are automatically dropped before any of the packet is presented to the AHB, thus reducing AHB activity
- Supports manual RX packet flush capabilities
- RX packet flush when there is lack of AHB resource

DMA Controller

The DMA uses separate transmit and receive lists of buffer descriptors, with each descriptor describing a buffer area in memory. This allows Ethernet packets to be broken up and scattered around the AHB memory space.

The DMA controller performs four types of operation on the AHB bus. In order of priority these are:

- Receive buffer manager write/read
- Transmit buffer manager write/read
- Receive data DMA write
- Transmit data DMA read

Transfer size is set to 32-bit words using the AHB bus width select bits in the Network Configuration register, and burst size may be programmed to single access or bursts of 4, 8, or 16 words using the DMA Configuration register.

Rx Buffers

Received frames, optionally including FCS, are written to receive AHB buffers stored in memory. The start location for each receive AHB buffer is stored in memory in a list of receive buffer descriptors at an address location pointed to by the receive-buffer queue pointer. The base address for the receive-buffer queue pointer is configured in software using the Receive Buffer Queue Base Address register.

Each list entry consists of two words. The first is the address of the receive AHB buffer and the second the receive status. If the length of a receive frame exceeds the AHB buffer length, the status word for the used buffer is written with zeroes except for the *start of frame* bit, which is always set for the first buffer in a frame. Bit zero of the address field is written to 1 to show that the buffer has been used. The receive-buffer manager then reads the location of the next receive AHB buffer and fills that with the next part of the received frame data. AHB buffers are filled until the frame is complete and the final buffer descriptor status word contains the complete frame status. Refer to [Table 16-2](#) for details of the receive buffer descriptor list.

Each receive AHB buffer start location is a word address. The start of the first AHB buffer in a frame can be offset by up to three bytes depending on the value written to bits [14] and [15] of the Network Configuration register. If the start location of the AHB buffer is offset the available length of the first AHB buffer is reduced by the corresponding number of bytes.

Table 16-2: Rx Buffer Descriptor Entry

Bit	Function
Word 0	
31:2	Address of beginning of buffer.
1	Wrap - marks last descriptor in receive buffer descriptor list.
0	Ownership - needs to be zero for the controller to write data to the receive buffer. The controller sets this to 1 once it has successfully written a frame to memory. Software must clear this bit before the buffer can be used again.

Table 16-2: Rx Buffer Descriptor Entry (Cont'd)

Bit	Function
Word 1	
31	Global all ones broadcast address detected.
30	Multicast hash match.
29	Unicast hash match.
28	External address match.
27	Reserved.
26:25	Specific address register match. Encoded as follows: 00b: Specific address register 1 match 01b: Specific address register 2 match 10b: Specific address register 3 match 11b: Specific address register 4 match If more than one specific address is matched only one is indicated with priority 4 down to 1.
24	This bit has a different meaning depending on whether RX checksum offloading is enabled. <ul style="list-style-type: none"> With RX checksum offloading disabled: (bit [24] clear in Network Configuration) Type ID register match found, bit [22] and bit [23] indicate which type ID register causes the match. With RX checksum offloading enabled: (bit [24] set in Network Configuration) 0b: The frame was not SNAP encoded and/or had a VLAN tag with the CFI bit set. 1b: The frame was SNAP encoded and had either no VLAN tag or a VLAN tag with the CFI bit not set.
23:22	This bit has a different meaning depending on whether RX checksum offloading is enabled. With RX checksum offloading disabled: (bit [24] clear in Network Configuration) Type ID register match. Encoded as follows: 00b: Type ID register 1 match 01b: Type ID register 2 match 10b: Type ID register 3 match 11b: Type ID register 4 match If more than one Type ID is matched only one is indicated with priority 4 down to 1. With RX checksum offloading enabled: (bit [24] set in Network Configuration) 00b: Neither the IP header checksum nor the TCP/UDP checksum was checked. 01b: The IP header checksum was checked and was correct. Neither the TCP or UDP checksum was checked. 10b: Both the IP header and TCP checksum were checked and were correct. 11b: Both the IP header and UDP checksum were checked and were correct.
21	VLAN tag detected – type ID of 0x8100. For packets incorporating the stacked VLAN processing feature, this bit is set if the second VLAN tag has a type ID of 0x8100.
20	Priority tag detected – type ID of 0x8100 and null VLAN identifier. For packets incorporating the stacked VLAN processing feature, this bit is set if the second VLAN tag has a type ID of 0x8100 and a null VLAN identifier.
19:17	VLAN priority – only valid if bit [21] is set.
16	Canonical format indicator (CFI) bit – only valid if bit 21 is set.
15	End of frame – when set the buffer contains the end of a frame. If end of frame is not set, then the only valid status bit is start of frame (bit 14).
14	Start of frame – when set the buffer contains the start of a frame. If both bits 15 and 14 are set, the buffer contains a whole frame.

Table 16-2: Rx Buffer Descriptor Entry (Cont'd)

Bit	Function
13	<p>This bit has a different meaning depending on whether ignore FCS mode are enabled. This bit is zero if ignore FCS mode is disabled.</p> <p>With ignore FCS mode enabled: (bit [26] set in Network Configuration Register). This indicates per frame FCS status as follows:</p> <ul style="list-style-type: none"> 0b: Frame had good FCS. 1b: Frame had bad FCS, but was copied to memory as ignore FCS enabled.
12:0	<p>These bits represent the length of the received frame which might or might not include FCS depending on whether FCS discard mode is enabled.</p> <ul style="list-style-type: none"> • With FCS discard mode disabled: (bit [17] clear in Network Configuration Register) Least significant 12-bits for length of frame including FCS. • With FCS discard mode enabled: (bit [17] set in Network Configuration Register) Least significant 12-bits for length of frame excluding FCS.

The start location of the receive-buffer descriptor list must be written with the receive-buffer queue base address before reception is enabled (receive enable in the Network Control register). Once reception is enabled, any writes to the Receive-buffer Queue Base Address register are ignored. When read, it returns the current pointer position in the descriptor list, though this is only valid and stable when receive is disabled.

If the filter block indicates that a frame should be copied to memory, the receive data DMA operation starts writing data into the receive buffer. If an error occurs, the buffer is recovered.

An internal counter represents the receive-buffer queue pointer and it is not visible through the CPU interface. The receive-buffer queue pointer increments by two words after each buffer has been used. It re-initializes to the receive-buffer queue base address if any descriptor has its wrap bit set.

As receive AHB buffers are used, the receive AHB buffer manager sets bit zero of the first word of the descriptor to logic one indicating the AHB buffer has been used.

Software should search through the "used" bits in the AHB buffer descriptors to find out how many frames have been received, checking the start of frame and end of frame bits.

Received frames are written out to the AHB buffers as soon as enough frame data exists in the packet buffer. This might mean that several full AHB buffers are used before some error conditions can be detected. If a receive error is detected the receive buffer currently being written is recovered. Previous buffers are not recovered. For example, when receiving frames with CRC errors or excessive length, it is possible that a frame fragment might be stored in a sequence of AHB receive buffers. Software can detect this by looking for the start of frame bit set in a buffer following a buffer with no end of frame bit set.

For a properly working 10/100/1000 Ethernet system there should be no excessive length frames or frames greater than 128 bytes with CRC errors. Collision fragments are less than 128 bytes long, therefore it is a rare occurrence to find a frame fragment in a receive AHB buffer, when using the default value of 128 bytes for the receive buffers size.

Only good received frames are written out of the DMA, so no fragments exist in the AHB buffers due to MAC receiver errors. There is still the possibility of fragments due to DMA errors, for example used bit read on the second buffer of a multi-buffer frame.

If bit zero of the receive buffer descriptor is already set when the receive buffer manager reads the location of the receive AHB buffer, then the buffer has been already used and cannot be used again until software has processed the frame and cleared bit zero. In this case, the “buffer not available” bit in the Receive Status register is set and an interrupt is triggered. The receive resource error statistics register is also incremented.

The user can optionally select whether received frames should be automatically discarded when no AHB buffer resource is available. This feature is selected via bit [24] of the DMA Configuration register (by default, the received frames are not automatically discarded). If this feature is off, then received packets remain stored in the packet buffer until an AHB buffer resource next becomes available. This can lead to an eventual packet buffer overflow if packets continue to be received when bit zero (used bit) of the receive-buffer descriptor remains set. Note that after a used bit has been read, the receive-buffer manager re-reads the location of the receive buffer descriptor every time a new packet is received.

A receive overrun condition occurs when the receive packet buffer is full, or because hresp was not okay. In all other modes, a receive overrun condition occurs when either the AHB bus was not granted quickly enough, or because hresp was not okay, or because a new frame has been detected by the receive block, but the status update or write back for the previous frame has not yet finished. For a receive overrun condition, the receive overrun interrupt is asserted and the buffer currently being written is recovered. The next frame that is received whose address is recognized reuses the buffer.

A write to bit [18] of the Network Control register forces a packet from the receive packet buffer to be flushed. This feature is only acted upon when the RX DMA is not currently writing packet data out to AHB – i.e., it is in an IDLE state. If the RX DMA is active, a write to this bit is ignored.

Tx Buffers

Frames to transmit are stored in one or more transmit AHB buffers. Transmit frames can be between 1 and 16,384 bytes long, so it is possible to transmit frames longer than the maximum length specified in the IEEE 802.3 standard. It should be noted that zero length AHB buffers are allowed and that the maximum number of buffers permitted for each transmit frame is 128.

The start location for each transmit AHB buffer is stored in memory in a list of transmit buffer descriptors at a location pointed to by the transmit-buffer queue pointer. The base address for this queue pointer is set in software using the Transmit-buffer Queue Base Address register. Each list entry consists of two words. The first is the byte address of the transmit buffer and the second containing the transmit control and status. For the packet buffer DMA, the start location for each AHB buffer is a byte address, the bottom bits of the address being used to offset the start of the data from the data-word boundary. For the FIFO based DMA, the address of the buffer is also a byte address. Frames can be transmitted with or without automatic CRC generation. If CRC is automatically generated, pad will also be automatically generated to take frames to a minimum length of 64 bytes. When CRC is not automatically generated (as defined in word 1 of the transmit buffer descriptor or through the control bus of the FIFO), the frame is assumed to be at least 64 bytes long and pad is not generated.

To transmit frames, the buffer descriptors must be initialized by writing an appropriate byte address to bits [31:0] in the first word of each descriptor list entry.

The second word of the transmit-buffer descriptor is initialized with control information that indicates the length of the frame, whether or not the MAC is to append CRC, and whether the buffer is the last buffer in the frame.

After transmission the status bits are written back to the second word of the first buffer along with the used bit. Bit [31] is the used bit which must be zero when the control word is read if transmission is to take place. It is written to one when the frame has been transmitted. Bits[29:20] indicate various transmit error conditions. Bit [30] is the wrap-bit which can be set for any buffer within a frame. If no wrap bit is encountered the queue pointer continues to increment.

The Transmit-buffer Queue Base Address register can only be updated while transmission is disabled or halted; otherwise any attempted write is ignored. When transmission is halted the transmit-buffer queue pointer maintains its value. Therefore, when transmission is restarted the next descriptor read from the queue is from immediately after the last successfully transmitted frame. While transmit is disabled (bit [3] of the network control is set Low), the transmit-buffer queue pointer resets to point to the address indicated by the Transmit-buffer Queue Base Address register. Note that disabling receive does not have the same effect on the receive-buffer queue pointer.

When the transmit queue is initialized, transmit is activated by writing to the transmit start bit (bit [9]) of the Network Control register. Transmit is halted when a buffer descriptor with its used bit set is read, a transmit error occurs, or by writing to the transmit halt bit of the Network Control register. Transmission is suspended if a pause frame is received while the pause enable bit is set in the network configuration register. Rewriting the start bit while transmission is active is allowed. This is implemented with a tx_go variable which is readable in the Transmit Status register at bit location 3.

The tx_go variable is reset when:

- Transmit is disabled
- A buffer descriptor with its ownership bit set is read
- Bit [10], tx_halt, of the Network Control register is written
- There is a transmit error such as too many retries, late collision (gigabit mode only) or a transmit under-run

To set tx_go, write to bit [9], tx_start, of the Network Control register. Transmit halt does not take effect until any ongoing transmit finishes.

The entire contents of the frame are read into the transmit packet buffer memory, so the retry attempt is replayed directly from the packet buffer memory rather than having to re-fetch through the AHB.

If a used bit is read mid way through transmission of a multi buffer frame this is treated as a transmit error. Transmission stops, tx_er is asserted and the FCS is bad.

If transmission stops due to a transmit error or a used bit being read, transmission is restarted from the first buffer descriptor of the frame being transmitted when the transmit start bit is rewritten.

Refer to [Table 16-3](#) for details of the transmit buffer descriptor list.

Table 16-3: Tx Buffer Descriptor Entry

Bit	Function
Word 0	
31:0	Byte address of buffer.
Word 1	
31	Used – must be zero for the controller to read data to the transmit buffer. The controller sets this to one for the first buffer of a frame once it has been successfully transmitted. Software must clear this bit before the buffer can be used again.
30	Wrap – marks last descriptor in transmit buffer descriptor list. This can be set for any buffer within the frame.
29	Retry limit exceeded, transmit error detected.
28	Always set to 0.
27	Transmit frame corruption due to AHB error – set if an error occurs whilst midway through reading transmit frame from the AHB, including HRESP errors and buffers exhausted mid frame (if the buffers run out during transmission of a frame then transmission stops, FCS shall be bad and tx_er asserted).
26	Late collision, transmit error detected. Late collisions only force this status bit to be set in gigabit mode.
25:23	Reserved.
22:20	Transmit IP/TCP/UDP checksum generation offload errors: <ul style="list-style-type: none"> • 000b: No Error. • 001b: The Packet was identified as a VLAN type, but the header was not fully complete, or had an error in it. • 010b: The Packet was identified as a SNAP type, but the header was not fully complete, or had an error in it. • 011b: The Packet was not of an IP type, or the IP packet was invalidly short, or the IP was not of type IPv4/IPv6. • 100b: The Packet was not identified as VLAN, SNAP or IP. • 101b: Non supported packet fragmentation occurred. For IPv4 packets, the IP checksum was generated and inserted. • 110b: Packet type detected was not TCP or UDP. TCP/UDP checksum was therefore not generated. For IPv4 packets, the IP checksum was generated and inserted. • 111b: A premature end of packet was detected and the TCP/UDP checksum could not be generated.
19:17	Reserved.
16	No CRC to be appended by MAC. When set this implies that the data in the buffers already contains a valid CRC and hence no CRC or padding is to be appended to the current frame by the MAC. This control bit must be set for the first buffer in a frame and is ignored for the subsequent buffers of a frame. Note that this bit must be clear when using the transmit IP/TCP/UDP checksum generation offload, otherwise checksum generation and substitution does not occur.
15	Last buffer, when set this bit indicates that the last buffer in the current frame has been reached.
14	Reserved.
13:0	Length of buffer.

DMA Bursting on the AHB

The DMA always uses SINGLE, or INCR type AHB accesses for buffer management operations. When performing data transfers, the AHB burst length used can be programmed using bits [4:0] of the DMA Configuration register so that either SINGLE, INCR or fixed length incrementing bursts (INCR4, INCR8 or INCR16) are used where possible.

When there is enough space and enough data to be transferred, the programmed fixed length bursts are used. If there is not enough data or space available, for example when at the beginning or the end of a buffer, SINGLE type accesses are used. SINGLE type accesses are also used at 1,024 byte boundaries, so that the 1 KB boundaries are not burst over per AHB requirements.

The DMA does terminate a fixed length burst early, unless an error condition occurs on the AHB or if receive or transmit are disabled in the Network Control register.

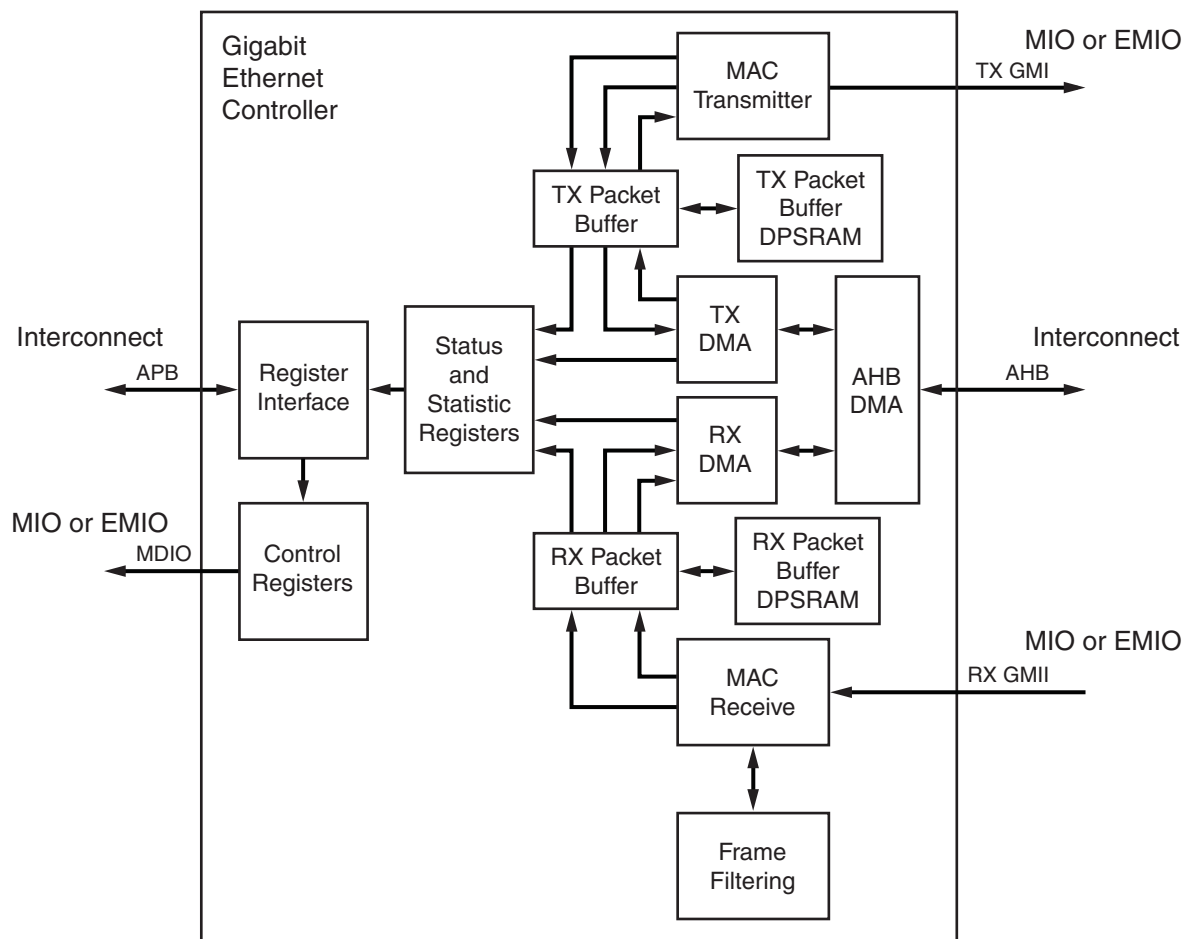
DMA Packet Buffer

The DMA uses packet buffers for both transmit and receive paths. This mode allows multiple packets to be buffered in both transmit and receive directions. This allows the DMA to withstand far greater access latencies on the AHB and make more efficient use of the AHB bandwidth.

Full packets are buffered which provides the opportunity to:

- Discard packets with error on the receive path before they are partially written out of the DMA thus saving AHB bus bandwidth and driver processing overhead
- Retry collided transmit frames from the buffer, thus saving AHB bus bandwidth,
- Implement transmit IP/TCP/UDP checksum generation offload.

With the packet buffers included, the structure of the controller data paths is as shown in [Table 16-3](#).



UG585_c16_03_071112

Figure 16-3: DMA Packet Buffer

In the transmit direction, the DMA continues to fetch packet data up to a limit of 256 packets, or until the buffer is full. The size of the buffer has a maximum usable size of 4 KB.

In the receive direction, if the buffer becomes full, then an overflow occurs. An overflow also occurs if the limit of 256 packets is breached. The size of the external buffer has a maximum usable size of 4 KB.

Tx Packet Buffer

The transmitter packet buffer continues to attempt to fetch frame data from the AHB system memory until the packet buffer itself is full, at which point it attempts to maintain its full level.

To accommodate the status and statistics associated with each frame, three words per packet are reserved at the end of the packet data. If the packet was bad and requires to be dropped, the status and statistics are the only information held on that packet. Storing the status in the DPRAM is required to decouple the DMA interface of the buffer from the MAC interface, to update the MAC status/stats, and to generate interrupts in the order in which the packets that they represent were fetched from the AHB memory.

If any errors occur on the AHB while reading the transmit frame, the fetching of packet data from AHB memory is halted. The MAC transmitter continues to fetch packet data, thereby emptying the packet buffer and allowing any good non-errored frames to be transmitted successfully. When these have been fully transmitted, the status/stats for the errored frame is updated and software is informed via an interrupt that an AHB error occurred. This way, the error is reported in the correct packet order.

The transmit packet buffer only attempts to read more frame data from the AHB when space is available in the packet buffer memory. If space is not available it must wait until the packet fetched by the MAC completes transmission and is subsequently removed from the packet buffer memory. Note that if full store and forward mode is active, and if a single frame is fetched that is too large for the packet buffer memory, the frame is flushed and the DMA is halted with an error status. This is because a complete frame must be written into the packet buffer before transmission can begin, and therefore the minimum packet buffer memory size should be chosen to satisfy the maximum frame to be transmitted in the application.

When the complete transmit frame is written into the packet buffer memory, a trigger is sent across to the MAC transmitter, which then begins reading the frame from the packet buffer memory. Because the whole frame is present and stable in the packet buffer memory, an underflow of the transmitter is not possible. The frame is kept in the packet buffer until notification is received from the MAC that the frame data has either been successfully transmitted or can no longer be re-transmitted (too many retries in half duplex mode). When this notification is received the frame is flushed from memory to make room for a new frame to be fetched from AHB system memory.

In half duplex mode, the frame is kept in the packet buffer until notification is received from the MAC that the frame data has either been successfully transmitted or can no longer be re-transmitted (too many retries in half duplex mode). When this notification is received the frame is flushed from memory to make room for a new frame to be fetched from AHB system memory.

In full duplex mode, the frame is removed from the packet buffer on-the-fly.

Other than underflow, the only MAC related errors that can occur are due to collisions during half duplex transmissions. When a collision occurs the frame still exists in the packet buffer memory so it can be retried directly from there. Only when the MAC transmitter has failed to transmit after sixteen attempts is the frame finally flushed from the packet buffer.

Rx Packet Buffer

The receive packet buffer stores frames from the MAC receiver along with their status and statistics. Frames with errors are flushed from the packet buffer memory, good frames are pushed onto the DMA AHB interface.

The receiver packet buffer monitors the FIFO writes from the MAC receiver and translates the FIFO pushes into packet buffer writes. At the end of the received frame the status and statistics are buffered so that the information can be used when the frame is read out. When programmed in full store and forward mode, if the frame has an error the frame data is immediately flushed from the packet buffer memory allowing subsequent frames to utilize the freed up space. The status and statistics for bad frames are still used to update the controller's registers.

Note: To accommodate the status and statistics associated with each frame, three words per packet are reserved at the end of the packet data. If the packet was bad and requires to be dropped, the status and statistics are the only information held on that packet.

The receiver packet buffer also detects a full condition such that an overflow condition can be detected. If this occurs, subsequent packets are dropped and an RX overflow interrupt is raised.

The DMA only begins packet fetches when the status and statistics for a frame are available. If the frame has a bad status due to a frame error, the status and statistics are passed onto the controller's registers. If the frame has a good status, the information is used to read the frame from the packet buffer memory and burst onto the AHB using the DMA buffer management protocol. After the last frame data has been transferred to the FIFO, the status and statistics are updated to the controller's registers.

16.2.6 Checksum Offloading

The controller can be programmed to perform IP, TCP, and UDP checksum offloading in both receive and transmit directions, enabled by setting bit [24] in the Network Configuration register for receive, and bit [11] in the DMA Configuration register for transmit.

IPv4 packets contain a 16-bit checksum field, which is the 16-bit 1's complement of the 1's complement sum of all 16-bit words in the header. TCP and UDP packets contain a 16-bit checksum field, which is the 16-bit 1's complement of the 1's complement sum of all 16-bit words in the header, the data, and a conceptual IP pseudo header.

To calculate these checksums in software requires each byte of the packet to be processed. For TCP and UDP this can use a large amount of processing power. Offloading the checksum calculation to hardware can result in significant performance improvements.

For IP, TCP, or UDP checksum offload to be useful, the operating system containing the protocol stack must be aware that this offload is available so that it can make use of the fact that the hardware can either generate or verify the checksum.

Rx Checksum Offload

When receive checksum offloading is enabled, the IPv4 header checksum is checked per RFC 791, where the packet meets the following criteria:

- If present, the VLAN header must be four octets long and the CFI bit must not be set
- Encapsulation must be RFC 894 Ethernet Type encoding or RFC 1042 SNAP encoding
- IPv4 packet
- IP header is of a valid length

The controller also checks the TCP checksum per RFC 793, or the UDP checksum per RFC 768, if the following criteria are met:

- IPv4 or IPv6 packet
- Good IP header checksum (if IPv4)
- No IP fragmentation
- TCP or UDP packet

When an IP, TCP, or UDP frame is received, the receive buffer descriptor provides an indication if the controller was able to verify the checksums. There is also an indication if the frame had SNAP encapsulation. These indication bits replace the type ID match indication bits when receive checksum offload is enabled.

If any of the checksums are verified to be incorrect by the controller, the packet is discarded and the appropriate statistics counter is incremented.

Tx Checksum Offload

The transmitter checksum offload is only available when the full store and forward mode is enabled. This is because the complete frame to be transmitted must be read into the packet buffer memory before the checksum can be calculated and written back into the headers at the beginning of the frame.

Transmitter checksum offload is enabled by setting bit [11] in the DMA Configuration register. When enabled, it monitors the frame as it is written into the transmitter packet buffer memory to automatically detect the protocol of the frame. Protocol support is identical to the receiver checksum offload.

For transmit checksum generation and substitution to occur, the protocol of the frame must be recognized and the frame must be provided without the FCS field, by ensuring that bit [16] of the transmit descriptor word 1 is clear. If the frame data already had the FCS field, this would be corrupted by the substitution of the new checksum fields.

If these conditions are met, the transmit checksum offload engine calculates the IP, TCP, and UDP checksums as appropriate. When the full packet is completely written into packet buffer memory, the checksums are valid and the relevant DPRAM locations are updated for the new checksum fields as per standard IP/TCP and UDP packet structures.

If the transmitter checksum engine is prevented from generating the relevant checksums, bits [22:20] of the transmitter DMA writeback status are updated to identify the reason for the error. Note that the frame is still transmitted but without the checksum substitution, as typically the reason that the substitution did not occur was that the protocol was not recognized.

16.2.7 IEEE 1588 Time Stamp Unit

IEEE 1588 is a standard for precision time synchronization in local area networks. It works with the exchange of special precision time protocol (PTP) frames. The PTP messages can be transported over *IEEE 802.3/Ethernet, over Internet Protocol Version 4* or over *Internet Protocol Version 6* as described in the *annex of IEEE P1588.D2.1*.

The controller detects when the PTP event messages sync, delay_req, pdelay_req and pdelay_resp are transmitted and received.

Synchronization between master and slave clocks is a two stage process.

- First, the offset between the master and slave clocks is corrected by the master sending a sync frame to the slave with a follow up frame containing the exact time the sync frame was sent. Hardware assist modules at the master and slave side detect exactly when the sync frame was sent by the master and received by the slave. The slave then corrects its clock to match the master clock.
- Second, the transmission delay between the master and slave is corrected. The slave sends a delay request frame to the master which sends a delay response frame in reply. Hardware assist modules at the master and slave side detect exactly when the delay request frame was sent by the slave and received by the master. The slave then has enough information to adjust its clock to account for delay. For example, if the slave was assuming zero delay the actual delay is half the difference between the transmit and receive time of the delay request frame (assuming equal transmit and receive times) because the slave clock is lagging the master clock by the delay time already.

For hardware assist it is necessary to timestamp when sync and delay_req messages are sent and received. The timestamp is taken when the message timestamp point passes the clock timestamp point. The message timestamp point is the SFD and the clock timestamp point is the MII interface. (The 1588 spec refers to sync and delay_req messages as event messages as these require timestamping. Follow up, delay response and management messages do not require timestamping and are referred to as general messages.)

1588 version 2 defines two additional PTP event messages. These are the peer delay request (Pdelay_Req) and peer delay response (Pdelay_Resp) messages. These messages are used to calculate the delay on a link. Nodes at both ends of a link send both types of frames (regardless of whether they contain a master or slave clock). The Pdelay_Resp message contains the time at which a Pdelay_Req was received and is itself an event message. The time at which a Pdelay_Resp message is received is returned in a Pdelay_Resp_Follow_Up message.

1588 version 2 introduces transparent clocks of which there are two kinds, peer-to-peer (P2P) and end-to-end (E2E). There is no transparent clock support.

The controller recognizes four different encapsulations for PTP event messages:

- 1588 version 1 (UDP/IPv4 multicast)

- 1588 version 2 (UDP/IPv4 multicast)
- 1588 version 2 (UDP/IPv6 multicast)
- 1588 version 2 (Ethernet multicast)

Note: Only multicast packets are supported.

The TSU consists of a timer and registers to capture the time at which PTP event frames cross the message timestamp point. These are accessible through the APB interface. An interrupt is issued when a capture register is updated.

The MAC provides timestamp registers that capture the departure time (for transmit) or arrival time (for receive) of PTP event packets (sync and delay request) and peer event packets (peer delay request or peer delay response). Interrupts are optionally generated upon timestamp capture.

The MAC also provides an option to timestamp all received packets by replacing the packet's FCS word with the nanoseconds portion of the timestamp. This eliminates the need to respond to received timestamp interrupts and to associate the timestamps with the correct received packets.

The timestamp unit includes a 62-bit timer counter. The lower 30 bits count nanoseconds and the upper 32 bits count seconds. Every clock cycle the counter is incremented by a programmable number of nanoseconds, and a mechanism is provided to handle fractional values. For example, at 120 MHz, the clock period is 8.333 ns. Every 3 clock cycles the counter is incremented twice by 8 and once by 9, for an average increment of 8.333 ns. The counter is clocked by the CPU 1x which is derived from the CPU clock.

There are six additional registers that capture the time at which PTP event frames are transmitted and received. An interrupt is issued when these registers are updated.

1588 Limitations

These topics can be addressed by software, but the added software workload limits the capabilities and accuracy of the IEEE1588 support. In many non-real-time operating systems, the interrupt response time is long, making it more difficult to achieve high accuracy. An estimate of achievable accuracy in a real-time environment is 10 μ s. In non-real time environments (e.g. Linux) the achievable accuracy is estimated at 100 – 1,000 μ s.

Time Counter Clock Input

The 62-bit time counter is clocked by the CPU_1x clock and there is no option for a separate clock input. Thus the choice of clock frequency and precision is related to the CPU clock frequency.

62-bit Time Counter Accuracy

The counter accuracy is limited to 62 bits. The least significant bits are in nanosecond units, and there is no direct support for counting fractions of nanoseconds. A mechanism is provided to allow fractional increments by averaging between two integer values, but its accuracy is limited and it creates jitter of up to ± 128 ns.

Counter Value to the PL

The 62-bit counter value is only accessible by reading a register. It is therefore not directly possible to schedule hardware events upon the counter reaching a specific value.

One Pulse per Second Output

Because there is no hardware access to the counter value, it is not possible to provide a 1 pps signal that is commonly used for lab tests of the synchronization accuracy.

Event Scheduling

The MAC does not provide event scheduling capability such as generating an interrupt upon the counter reaching a specific value.

Synchronizing the Two Ethernet Controllers

The two Ethernet cores are completely independent, and there is no hardware mechanism provided for synchronizing the counter values of the two Ethernet cores, or for making the counter of one Ethernet core slave to the other.

Single Packet Queue — No Tagging

All received packets are written into the same queue (or packet buffer) in memory. Similarly, all transmitted packets are read from the same queue. There is no support for multiple queues. A single queue makes it more difficult for the software to associate (tag) a packet with a timestamp.

FIFO Depth

The timestamp registers are 1-deep, so new events overwrite old values. This requires the software to have a fast enough response time to avoid event overrun.

Designing the Timestamp Unit in the PL

Improvement in performance and accuracy of time stamping can be achieved by implementing the timestamp unit in the PL instead of using the built-in timestamp unit in the MAC. Using this approach, the time counter and the timestamp registers are implemented in the PL, and the PTP frame recognition remains in the Ethernet core.

The outputs from the controller can be used to implement the timestamp unit in the PL. The EMIO signals for this are listed in Table 10, *Ethernet GMII Interface Signals via EMIO Interface*. The following items are not supported:

- Support of unicast packets
- Support of transparent clocks
- Single packet queue – no tagging

Designing a PTP Packet Tagging and Capture Module

Designing a separate timestamp unit as described above addresses most of the items, but it does not address the hardware mechanism for tagging the PTP packets, i.e., associating the packets with their corresponding timestamp. For example, if along with the timestamp some identifying packet attribute were captured, it would allow software to easily associate the timestamp with the correct PTP event. Examples of useful packet attributes are packet identification or serial number, or the memory address it was read from or written to. It is also possible to capture the entire packet along with its timestamp (for both transmit and receive), and make it available to software, for example via FIFOs or via circular buffers in main memory. Such a function can be implemented in the PL along with the timestamp unit as described above.

However, implementation requires access to the packet data stream itself. In order to have access to the packet data stream, the controller needs to be pinned-out through the EMIO using GMII, instead of MIO. By selecting this option, the GMII signals are exposed to the PL and can be used to detect and capture the PTP packets. Note that it is still possible to use the PTP frame recognition in the MAC, or it is possible to design this function in the PL as well (e.g., if support for unicast packets is required).

16.2.8 MAC 802.3 Pause Frame Support

Note: See Clause 31, and Annex 31A and 31B of the *IEEE standard 802.3* for a full description of pause operation.

The start of an 802.3 pause frame is shown in [Table 16-4](#)

Table 16-4: Pause Frame Information

Destination Address	Source Address	Type (MAC Control Frame)	Pause Opcode	Pause Time
0x0180C2000001	6 bytes	0x8808	0x0001	2 bytes

The controller supports both hardware controlled pause of the transmitter upon reception of a pause frame and hardware generated pause frame transmission.

802.3 Pause Frame Reception

Bit [13] of the Network Configuration register is the pause enable control for reception. If this bit is set transmission pauses if a non zero pause quantum frame is received.

If a valid pause frame is received then the pause time register is updated with the new frame's pause time regardless of whether a previous pause frame is active or not. An interrupt (either bit [12] or bit [13] of the Interrupt Status register) is triggered when a pause frame is received, but only if the interrupt has been enabled (bit [12] and bit [13] of the Interrupt Mask register). Pause frames received with non zero quantum are indicated through the interrupt bit [12] of the Interrupt Status register. Pause frames received with zero quantum are indicated on bit [13] of the Interrupt Status register.

When the pause time register is loaded and the frame currently being transmitted has been sent, no new frames are transmitted until the pause time reaches zero. The loading of a new pause time, and

hence the pausing of transmission, only occurs when the controller is configured for full duplex operation. If the controller is configured for half duplex there is no transmission pause, but the pause frame received interrupt is still triggered. A valid pause frame is defined as having a destination address that matches either the address stored in Specific Address register 1 or if it matches the reserved address of 0x0180C2000001. It must also have the MAC control frame type ID of 0x8808 and have the pause opcode of 0x0001.

Pause frames that have FCS or other errors are treated as invalid and are discarded. 802.3 Pause frames that are received after priority based flow control (PFC) has been negotiated are also discarded. Valid pause frames received increment the Pause Frames Received Statistic register.

The Pause Time register decrements every 512 bit times once transmission has stopped. For test purposes, the retry test bit can be set (bit [12] in the Network Configuration register) which causes the Pause Time register to decrement every tx_clk cycle when transmission has stopped.

The interrupt (bit [13] in the Interrupt Status register) is asserted whenever the pause time register decrements to zero (assuming it has been enabled by bit [13] in the Interrupt Mask register). This interrupt is also set when a zero quantum pause frame is received.

802.3 Pause Frame Transmission

Automatic transmission of pause frames is supported through the transmit pause frame bits of the Network Control register and from the external input pins tx_pause, tx_pause_zero and tx_pfc_sel. If either bit [11] or bit [12] of the Network Control register is written with logic 1, or if the input signal tx_pause is toggled when tx_pfc_sel is Low, an 802.3 pause frame is transmitted providing full duplex is selected in the Network Configuration register and the transmit block is enabled in the Network Control register.

Pause frame transmission occurs immediately if transmit is inactive or if transmit is active between the current frame and the next frame due to be transmitted.

Transmitted pause frames comprise of the following:

- A destination address of 01-80-C2-00-00-01
- A source address taken from Specific Address register 1
- A type ID of 88-08 (MAC control frame)
- A pause opcode of 00-01
- A pause quantum register
- Fill of 00 to take the frame to minimum frame length
- Valid FCS

The pause quantum used in the generated frame depends on the trigger source for the frame as follows:

- If bit [11] is written with a one, the pause quantum is taken from the Transmit Pause Quantum register. The Transmit Pause Quantum register resets to a value of 0xFFFF giving maximum pause quantum as the default.
- If bit [12] is written with a one, the pause quantum is zero.

- If the tx_pause input is toggled, tx_pfc_sel is Low and the tx_pause_zero input is held Low until the next toggle, the pause quantum is taken from the Transmit Pause Quantum register.
- If the tx_pause input is toggled, tx_pfc_sel is Low and the tx_pause_zero input is held High until the next toggle, the pause quantum is zero.

After transmission, a pause frame transmitted interrupt is generated (bit [14] of the Interrupt Status register) and the only statistics register that is incremented is the Pause Frames Transmitted register.

Pause frames can also be transmitted by the MAC using normal frame transmission methods.

MAC PFC Priority Based Pause Frame Support

Note: Refer to the 802.1Qbb standard for a full description of priority based pause operation.

The controller supports PFC priority based pause transmission and reception. Before PFC pause frames can be received, bit 16 of the Network Control register must be set. The start of a PFC pause frame is shown in [Table 16-5](#).

Table 16-5: PFC Priority Based Pause Frame Info

Destination Address	Source Address	Type (MAC Control Frame)	Pause Opcode	Priority Enable Vector	Pause Times
0x0180C2000001	6 bytes	0x8808	0x0101	2 bytes	8 * 2 bytes

PFC Pause Frame Reception

The ability to receive and decode priority based pause frames is enabled by setting bit [16] of the Network Control register. When this bit is set, the controller matches either classic 802.3 pause frames or PFC priority based pause frames. Once a priority based pause frame has been received and matched, then from that moment on the controller only matches on priority based pause frames (this is an 802.1Qbb requirement, known as PFC negotiation). Once priority based pause has been negotiated, any received 802.3x format pause frames are not acted upon. The state of PFC negotiation is identified via the output pfc_negotiate.

If a valid priority based pause frame is received then the controller decodes the frame and determines which, if any, of the eight priorities require to be paused. Up to eight Pause Time registers are then updated with the eight pause times extracted from the frame, regardless of whether a previous pause operation is active or not. An interrupt (either bit [12] or bit [13] of the Interrupt Status register) is triggered when a pause frame is received, but only if the interrupt has been enabled (via bit[12] and bit[13] of the Interrupt Mask register).

Pause frames received with non zero quantum are indicated through the interrupt bit[12] of the Interrupt Status register. Pause frames received with zero quanta are indicated on bit[13] of the Interrupt Status register. The state of the eight pause time counters are indicated through the outputs rx_pfc_paused. These outputs remain High for the duration of the pause time quanta. The loading of a new pause time only occurs when the controller is configured for full duplex operation. If the controller is configured for half duplex, the pause time counters are not loaded, but the pause frame received interrupt is still triggered.

A valid pause frame is defined as having a destination address that matches either the address stored in Specific Address register 1 or if it matches the reserved address of 0x0180C2000001. It must also have the MAC control frame type ID of 0x8808 and have the pause opcode of 0x0101.

Pause frames that have FCS or other errors are treated as invalid and are discarded. Valid pause frames received increment the pPause Frames Received Statistic register.

The Pause Time registers decrement every 512 bit times immediately following the PFC frame reception. For test purposes, the retry test bit can be set (bit [12] in the Network Configuration register) which causes the Pause Time register to decrement every rx_clk cycle once transmission has stopped.

The interrupt (bit [13] in the Interrupt Status register) is asserted whenever the Pause Time register decrements to zero (assuming it has been enabled by bit [13] in the Interrupt Mask register). This interrupt is also set when a zero quantum pause frame is received.

PFC Pause Frame Transmission

Automatic transmission of pause frames is supported through the transmit priority based pause frame bit of the Network Control register and from the external input pins tx_pause, tx_pfc_pause[7:0], tx_pfc_pause_zero[7:0], and tx_pfc_sel. If bit 17 of the Network Control register is written with logic 1, or if the input signal tx_pause is toggled when tx_pfc_sel is High, a PFC pause frame is transmitted providing full duplex is selected in the Network Configuration register and the transmit block is enabled in the Network Control register. When bit 17 of the Network Control register is set, the fields of the priority based pause frame are built using the values stored in the Transmit PFC Pause register.

Pause frame transmission occurs immediately if transmit is inactive or if transmit is active between the current frame and the next frame due to be transmitted.

Transmitted pause frames comprise of the following:

- A destination address of 01-80-C2-00-00-01
- A source address taken from Specific Address register 1
- A type ID of 88-08 (MAC control frame)
- A pause opcode of 01-01
- A priority enable vector taken from tx_pfc_pause or the Transmit PFC Pause register
- Eight pause quantum registers
- Fill of 00 to take the frame to minimum frame length
- Valid FCS

The Pause Quantum registers used in the generated frame depend on the trigger source for the frame as follows:

- If bit [17] of the Network Control register is written with a one then the priority enable vector of the priority based pause frame is set equal to the value stored in the Transmit PFC Pause register [7:0]. For each entry equal to zero in the Transmit PFC Pause register[15:8], the pause quantum field of the pause frame associated with that entry is taken from the Transmit Pause Quantum register. For each entry equal to one in the Transmit PFC Pause register[15:8], the pause quantum associated with that entry is zero.

- The Transmit Pause Quantum register resets to a value of 0xFFFF giving maximum pause quantum as default.
- If the tx_pause input is toggled and tx_pfc_sel is High then the priority enable vector of the priority based pause frame is set equal to the value in tx_pfc_pause [7:0]. For each entry equal to zero in tx_pfc_pause_zero [7:0], the pause quantum field of the pause frame associated with that entry is taken from the Transmit Pause Quantum register. For each entry equal to one in tx_pfc_pause_zero [7:0], the pause quantum associated with that entry is zero.

After transmission, a pause frame transmitted interrupt is generated (bit 14 of the Interrupt Status register) and the only statistics register that is incremented is the Pause Frames Transmitted register.

PFC Pause frames can also be transmitted by the MAC using normal frame transmission methods.

16.3 Programming Examples

16.3.1 Initialization

Initialization/configuration (e.g. loopback mode, frequency ratios) must be done while the transmit and receive circuits are disabled. See the description of the Network Control register and Network Configuration register in [Appendix B, Register Details](#). The following sequence of operations must be followed to change loopback mode:

- Write to the Network Control register to disable transmit and receive circuits.
- Write to the Network Control register to change loopback mode.
- Write to the Network Control register to re-enable transmit or receive circuits.

Note: These writes to the Network Control register cannot be combined in any way.

Receive Buffer List

Receive data is written to areas of data (i.e., buffers) in system memory. These buffers are listed in another data structure that also resides in main memory. This data structure (receive buffer queue) is a sequence of descriptor entries.

The Receive-buffer Queue Pointer register points to this data structure as shown in [Figure 16-4](#).

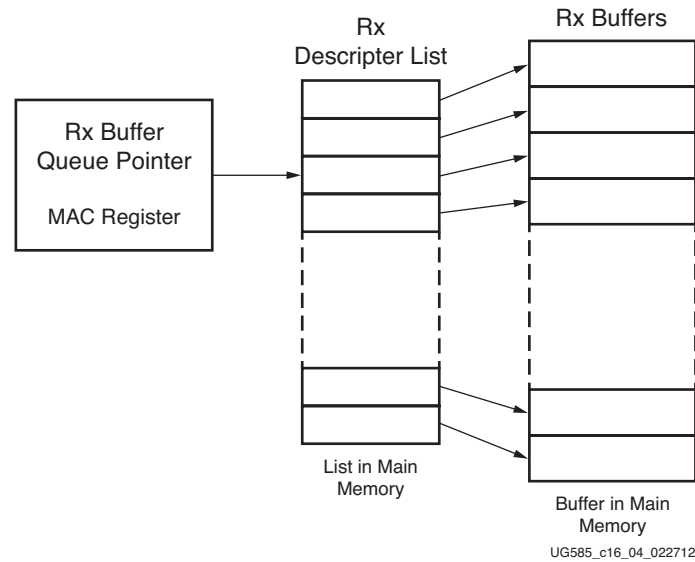


Figure 16-4: Rx Buffer Queue Structure

To create this list of buffers:

1. Allocate a number (N) of buffers of X bytes in system memory, where X is the DMA buffer length programmed in the DMA Configuration register.
2. Allocate an area 8N bytes for the receive buffer descriptor list in system memory and create N entries in this list. Mark all entries in this list as owned by controller, i.e., bit[0] of word 0 set to 0.
3. Mark the last descriptor in the queue with the wrap bit (bit[1] in word 0 set to 1).
4. Write address of receive buffer descriptor list and control information to controller register receive buffer queue pointer.
5. The receive circuits can then be enabled by writing to the Address Recognition registers and the Network Control register.

Tx Buffer List

Transmit data is read from areas of data (the buffers) in system memory. These buffers are listed in another data structure that also resides in main memory. This data structure (Transmit Buffer Queue) is a sequence of descriptor entries as defined in [Table 16-3](#).

The Transmit Buffer Queue Pointer register points to this data structure.

To create this list of buffers:

1. Allocate a number (N) of buffers of between 1 and 2,047 bytes of data to be transmitted in system memory. Up to 128 buffers per frame are allowed.
2. Allocate an area 8N bytes for the transmit buffer descriptor list in system memory and create N entries in this list. Mark all entries in this list as owned by controller, i.e., bit[31] of word 1 set to 0.
3. Mark the last descriptor in the queue with the wrap bit (bit[30] in word 1 set to 1).

4. Write address of transmit buffer descriptor list and control information to controller register transmit buffer queue pointer.
5. The transmit circuits can then be enabled by writing to the Network Control register.

Address Matching

The controller register pair hash address and the four specific address register-pairs must be written with the required values. Each register pair comprises of a bottom register and top register with the bottom register being written first. The address matching is disabled for a particular register pair after the bottom register has been written and re-enabled when the top register is written. Each register pair may be written at any time, regardless of whether the receive circuits are enabled or disabled.

As an example, to set Specific Address register 1 to recognize destination address 21:43:65:87:A9:CB, the following values would be written to Specific Address register 1 bottom and Specific Address register 1 top:

- Specific Address register 1 bottom bits 31:0 (0x98): 8765_4321 hex.
- Specific Address register 1 top bits 31:0 (0x9C): 0000_cba9 hex.

16.3.2 PHY Maintenance

The PHY Maintenance register is implemented as a shift register. Writing to the register starts a shift operation which is signaled as complete when bit two is set in the Network Status register (about 2,000 CPU_1x clock cycles later when bits [18:16] are set to 010 in the Network Configuration register). An interrupt is generated as this bit is set.

During this time, the MSB of the register is output on the mdio_in pin and the LSB updated from the mdio_in pin with each MDC cycle. This causes the transmission of a PHY management frame on MDIO. See *Section 22.2.4.5 of the IEEE 802.3 standard*.

Reading during the shift operation returns the current contents of the shift register. At the end of the management operation the bits are shifted back to their original locations. For a read operation the data bits are updated with data read from the PHY. It is important to write the correct values to the register to ensure that a valid PHY management frame is produced.

The MDC should not toggle faster than 2.5 MHz (minimum period of 400 ns), as defined by the IEEE 802.3 standard. Mdc is generated by dividing down CPU_1x clock. Three bits in the network configuration register determine by how much CPU_1x clock should be divided to produce MDC. The default is 32 which is acceptable for a CPU_1x clock running up to 80 MHz.

16.3.3 Servicing Interrupts

There are 18 interrupt conditions that are detected within the controller. These are OR-ed to make a single interrupt (or multiple interrupts if priority queuing is enabled). Depending on the overall system design this might be passed through a further level of interrupt collection (interrupt controller). On receipt of the interrupt signal, the CPU enters the interrupt handler and reads the

Interrupt Status register to determine the cause. To clear the interrupt, a write-one-to-clear scheme is used. Write the Interrupt Status register interrupt bit with 1 to clear the interrupt.

At reset all interrupts are disabled. To enable an interrupt, write to the Interrupt Enable register with the pertinent interrupt bit set to 1. To disable an interrupt, write to Interrupt Disable register with the pertinent interrupt bit set to 1. To check whether an interrupt is enabled or disabled, read the Interrupt Mask register: if the bit is set to 1, the interrupt is disabled.

16.3.4 Transmitting Frames

To set up a frame for transmission:

1. Enable transmit in the Network Control register.
2. Allocate an area of system memory for transmit data. This does not have to be contiguous. Varying byte lengths can be used if they conclude on byte borders.
3. Set-up the transmit buffer list by writing buffer addresses to word zero of the transmit buffer descriptor entries and control and length to word one.
4. Write data for transmission into the buffers pointed to by the descriptors.
5. Write the address of the first buffer descriptor to transmit buffer descriptor queue pointer.
6. Enable appropriate interrupts.
7. Write to the transmit start bit in the Network Control register.

16.3.5 Receiving Frames

When a frame is received and the receive circuits are enabled, the controller checks the address and, in the following cases, the frame is written to system memory:

- If it matches one of the four specific address registers.
- If it matches one of the four type ID registers.
- If it matches the hash address function.
- If it is a broadcast address (0xFFFFFFFF) and broadcasts are allowed.
- If the controller is configured to "copy all frames".
- If a match is found in the external address filtering interface.

The register receive buffer queue pointer points to the next entry in the receive buffer descriptor list and the controller uses this as the address in system memory to write the frame to.

Once the frame has been completely and successfully received and written to system memory, the controller then updates the receive buffer descriptor entry with the reason for the address match and marks the area as being owned by software. Once this is complete, a receive-complete interrupt is set. Software is then responsible for copying the data to the application area and releasing the buffer (by writing the ownership bit back to 0).

If the controller is unable to write the data at a rate to match the incoming frame, then a receive overrun interrupt is set. If there is no receive buffer available, i.e. the next buffer is still owned by

software, a receive-buffer not available interrupt is set. If the frame is not successfully received, a Statistic register is incremented and the frame is discarded without informing software.

16.4 Register Overview

16.4.1 Control Registers

Control registers drive the management data input/output (MDIO) interface, set-up DMA activity, start frame transmission, and select the different modes of operation such as full duplex, half duplex and 10/100/1000 Mb/s operation. (See [Table 16-6](#).)

Table 16-6: Ethernet Control Register Overview

Function	Register Name	Description
MAC Configuration	net_{cfg,ctrl,status} tx_pauseq rx_pauseq tx_pfc_pause ipg_stretch stacked_vlan	Network control, configuration and status. Rx, Tx Pause clocks. IPG stretch.
DMA Unit	tx_status rx_status tx_qbar tx_qbar_q{1:7} rx_qbar rx_qbar_q{1:7} dma_cfg	Control. Receive, Transmit Status. Receive, Transmit Queue Base Address Control.
Interrupts	intr_{status,en,dis, mask}	Interrupt status, enable/disable, and mask
	intr_dis_pq{1:7}	
	intr_en_pq{1:7}	
	intr_mask_pq{1:7}	
	wake_on_lan	
	isr_pq{1:7}	
PHY Maintenance	phy_maint	PHY maintenance
MAC Address Filtering and ID Match	hash_{top,bot} spec_addr{1:4}_{bot,top} spec_addr1_mask_{bot,top} type_id_match{1:4}	Hash address, Specific {4:1} addresses High/Low. Match Type.

Table 16-6: Ethernet Control Register Overview (Cont'd)

Function	Register Name	Description
IEEE 1588 – Precision Time Protocol	timer_{s,ns} timer_{adjust,incr} timer_strobe_{s,ns}	IEEE 1588 second, nanosecond counter and adjustment, increment.
	ptp_tx_{s,ns} ptp_peer_tx_{s,ns}	IEEE 1588 Tx normal/peer second, nanosecond counter.
	ptp_rx_{s,ns} ptp_peer_rx_{s,ns}	IEEE 1588 Rx normal/peer second, nanosecond counter.
Clocks and Reset	slcr.GEM{1,0}_CLK_CTRL slcr.GEM{1,0}_RCLK_CTRL slcr.GEM{1,0}_CPU_1XCLKACT slcr.GEM_RST_CTRL	Refer to Chapter 25, Clocks and Chapter 26, Reset System for more information.
IO Signal Routing	slcr.MIO_FMIO_GEM_SEL slcr.MIO_PIN_{xxx}	Refer to IOP Interface Connections, page 40 for MIO pin programming information.

16.4.2 Status and Statistics Registers

The statistics registers hold counts for various types of events associated with transmit and receive operations. These registers, along with the status words stored in the receive buffer list, enable software to generate network management statistics compatible with IEEE 802.3. (See [Table 16-7](#)).

Table 16-7: Ethernet Status and Statistics Register Overview

Function	Hardware Register Name	Description
Frame Tx Statistics	frames_tx broadcast_frames_tx multi_frames_tx	Error-free Tx frame, pause frame counts and bytes counts.
	frames_64b_tx frames_65to127b_tx frames_128to255b_tx frames_256to511b_tx frames_512to1023b_tx frames_1024to1518b_tx	Error-free frames transmitted: totals by size.
	octets_tx_{top,bot}	Octets transmitted.
	deferred_tx_frames	
	pause_frames_tx tx_under_runs	Octets received High/Low, Under-run error count.
Frame Tx Statistics for Half Duplex Transmission	{multi,single}_collisn_frames excessive_collisns late_collisns carrier_sense_errs	Single/multiple frame, excessive/late collisions, deferred Tx frames, Tx carrier sense error counters.

Table 16-7: Ethernet Status and Statistics Register Overview (Cont'd)

Function	Hardware Register Name	Description
Frame Rx Statistics	frames_rx bdcast_fames_rx multi_frames_rx	Error-free frames received: normal, broadcast, multicast, pause.
	frames_64b_rx frames_65to127b_rx frames_128to255b_rx frames_256to511b_rx frames_512to1023b_rx frames_1024to1518b_rx	Error-free frames received: totals by size.
	{undersz,oversz,jab}_rx	Undersize, oversize and jabber frames.
	fcs_errors length_field_errors	Frame sequence, length, symbol, alignment error counters.
	octets_rx_{top,bot}	
	rx_symbol_errors align_errors rx_resource_errors rx_overrun_errors	Rx resource, overrun and last statistic clearing offset for clearing.
Frame Rx Checksum Error Statistics	ip_hdr_csum_errors tcp_csum_errors udp_csum_errors	Checksum error counters: IP Header, TCP, UDP

16.5 Signals and I/O Connections

16.5.1 MIO–EMIO Interface Routing

The I/O interface is routed to the MIO for RGMII, and to the EMIO for GMII/MII connectivity. The PL can modify the GMII/MII interface from the MAC to construct other Ethernet interfaces that connect to external devices via PL pins. The routing of the Ethernet communications signals are shown in Figure 16-5. The Ethernet communications ports are independently routed to the MIO pins (as RGMII) or to a set of EMIO interface signals (as GMII). When using the EMIO interface both the TX and RX clocks are inputs to the PS.

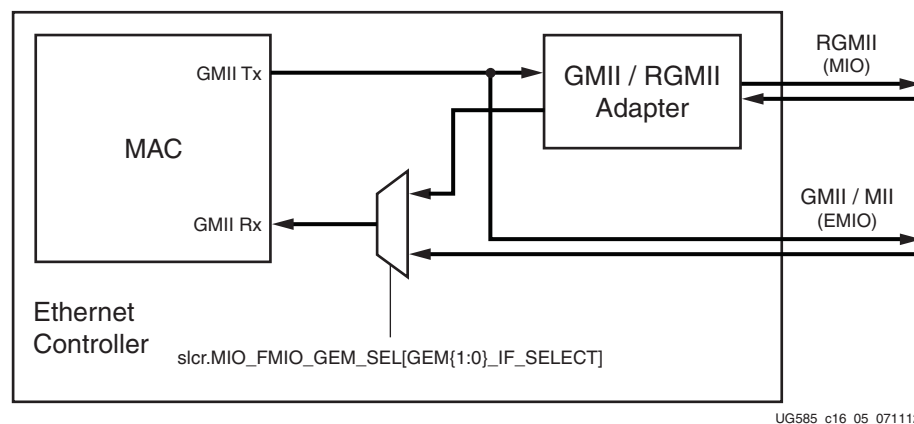


Figure 16-5: Interface Select Multiplexer

16.5.2 Precision Time Protocol

The PTP signals connected to the Ethernet controller provide the capability to handle IEEE-1588 precision time protocol (PTP) signaling.

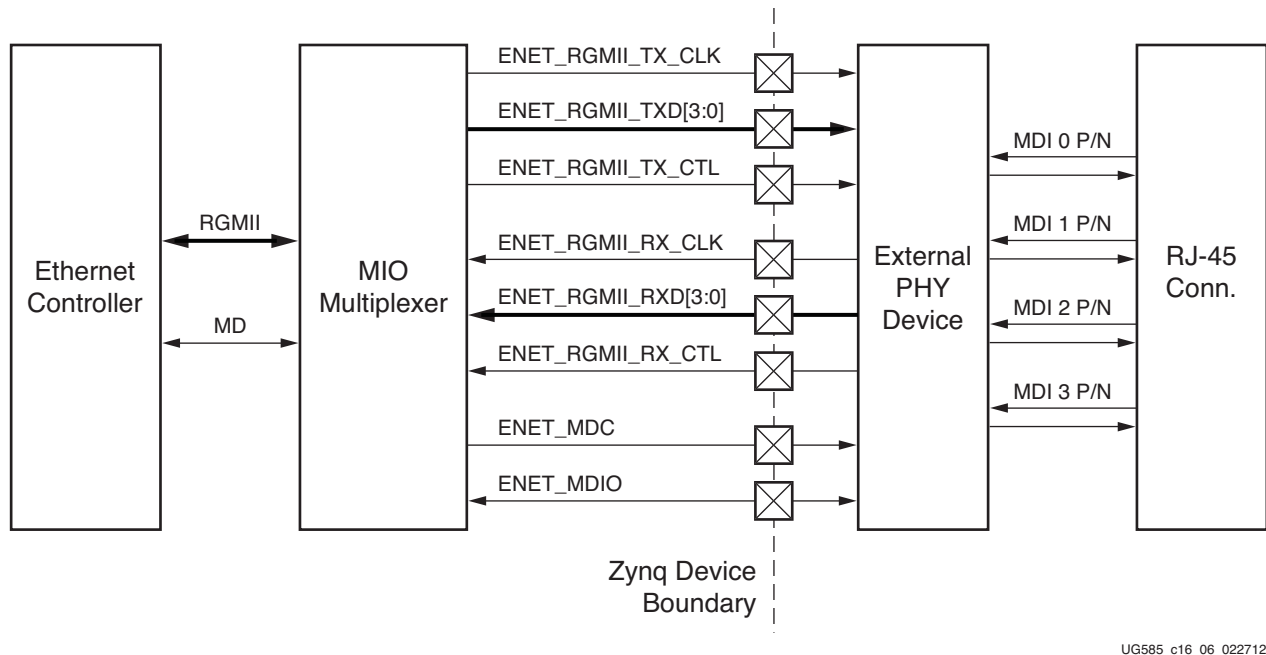
16.5.3 Programmable Logic (PL) Implementations

There are options to provide further external interface standard support by linking the GMII signals on the EMIO interface to the PL. Users can design and connect logic to generate other interface standards on the PL pins.

TBI support can be provided by connecting the GMII to a TBI compatible logic core in the PL which provides the PCS functions required for ten-bit interfacing to an external PHY via the PL pins. SGMII or 1000 Base-X support can be provided by connecting the GMII to an SGMII or 1000 Base-X compatible logic core which provides the required PCS functions and signal adaptation and drives an MGT for serial interfacing to an external PHY.

16.5.4 Wiring Diagram Example

An example Ethernet communications wiring connection is shown in [Figure 16-6](#)



UG585_c16_06_022712

Figure 16-6: Ethernet Communications Wiring Connections

16.5.5 RGMII Interface via MIO

All Ethernet I/O pins routed through the MIO are on MIO Bank 1 (see [Table 16-8](#)). The MIO pins and restrictions based on device version are shown in the MIO table in section [2.4.4 MIO-at-a-Glance Table](#).

Table 16-8: Ethernet RGMII Interface Signals via MIO Pins

Controller Signal		MIO Pins			
Signal Description	Default Controller Input Value	GigE 0	GigE 1	Name	I/O
Tx clock to PHY	~	16	28	RGMII_TX_CLK	O
Tx control to PHY	~	21	33	RGMII_TX_CTL	O
Tx data 0 to PHY	~	17	29	RGMII_TX_D0	O
Tx data 1 to PHY	~	18	30	RGMII_TX_D1	O
Tx data 2 to PHY	~	19	31	RGMII_TX_D2	O
Tx data 3 to PHY	~	20	32	RGMII_TX_D3	O
Rx clock from PHY	0	22	34	RGMII_RX_CLK	I
Rx control from PHY	0	27	39	RGMII_RX_CTL	I

Table 16-8: Ethernet RGMII Interface Signals via MIO Pins (Cont'd)

Controller Signal		MIO Pins			
Signal Description	Default Controller Input Value	GigE 0	GigE 1	Name	I/O
Rx data 0 from PHY	0	23	35	RGMII_RX_D0	I
Rx data 1 from PHY	0	24	36	RGMII_RX_D1	I
Rx data 2 from PHY	0	25	37	RGMII_RX_D2	I
Rx data 3 from PHY	0	26	38	RGMII_RX_D3	I

16.5.6 GMII/MII Interface via EMIO

Ethernet GMII/MII interface signals routed through the EMIO are identified in [Table 16-9](#).

Table 16-9: Ethernet GMII/MII Interface Signals via EMIO Interface

Interface Signal	Reference Clock	Default Controller Input Value	EMIO Interface Signals	
			Name	I/O
Carrier sense	~		EMIOENET[1,0]GMIICRS	I
Collision detect	~		EMIOENET[1,0]GMIICOL	I
Controller Interrupt input	~		EMIOENET[1,0]EXTINTIN	I
Tx Signals				
Tx Clock	~		EMIOENET[1,0]GMIITXCLK	I
Tx Data (7:0)	Tx Clk	~	EMIOENET[1,0]GMIITXD[7:0]	O
Tx Enable	Tx Clk	~	EMIOENET[1,0]GMIITXEN	O
Tx Error	Tx Clk	~	EMIOENET[1,0]GMIITXER	O
Tx Timestamp Signals				
Tx Start-of-Frame	Tx Clk	~	EMIOENET[1,0]SOFTX	O
Tx PTP delay req frame detected	Tx Clk	~	EMIOENET[1,0]PTPDELAYREQTX	O
Tx PTP peer delay frame detect	Tx Clk	~	EMIOENET[1,0]PTPPDELAYREQTX	O
Tx PTP pear delay response frame detected	Tx Clk	~	EMIOENET[1,0]PTPPDELAYRESPTX	O
Tx PTP sync frame detected	Tx Clk	~	EMIOENET[1,0]PTPSYNCFRAME TX	O
Rx Signals				
Rx Clock	~		EMIOENET[1,0]GMIIRXCLK	I
Rx Data (7:0)	Rx Clk		EMIOENET[1,0]GMIIRXD[7:0]	I
Rx Data valid	Rx Clk		EMIOENET[1,0]GMIIRXDV	I
Rx Error	Rx Clk		EMIOENET[1,0]GMIIRXER	I
Rx Timestamp Signals				
Rx Start of Frame	Rx Clk	~	EMIOENET[1,0]GMIISOFRX	O
Rx PTP delay req frame detected	Rx Clk	~	EMIOENET[1,0]PTPDELAYREQRX	O

Table 16-9: Ethernet GMII/MII Interface Signals via EMIO Interface (Cont'd)

Interface Signal	Reference Clock	Default Controller Input Value	EMIO Interface Signals	
			Name	I/O
Rx PTP peer delay frame detected	Rx Clk	~	EMIOENET[1,0]PTPPDELAYREQRX	O
Rx PTP peer delay response frame detected	Rx Clk	~	EMIOENET{0,1}PTPPDELAYRESPRX	O
Rx PTP sync frame detected	Rx Clk	~	EMIOENET{0,1}PTPSYNCFRAMERX	O

Notes:

1. If using MII connect the RX[7:4] bits to logic zero.

16.5.7 MDIO Interface Signals via MIO and EMIO

MDIO interface signals routed through the MIO and EMIO are identified in [Table 16-10](#).

Table 16-10: MDIO Interface Signals via MIO and EMIO

MDIO Interface	Default Controller Input Value	MIO Pins				EMIO Interface Signals	
		GigE 0	GigE 1	I/O	Name	Name	I/O
MD clock output	~	52	52	O	MDIO_CLK	EMIOENET[1:0]MDIOMDC	O
MD data output	~	53	53	IO	MDIO_IO	EMIOENET[1:0]MDIOO	O
MD data 3-state	~					EMIOENET[1:0]MDIOTN	O
MD data input	0					EMIOENET[1:0]MDIOI	I

SPI Controller

17.1 Introduction

The serial peripheral interface (SPI) bus controller module can function in master mode, slave mode or multi-master mode. The master device always initiates the data frame. Multiple slave devices are allowed with individual slave select (SS) lines. The SPI controller supports a 50 MHz maximum external clock rate if the signals are routed via the MIO. If the signals are routed via EMIO to the PL pins, the maximum external clock rate is 25 MHz. The controller supports 128-byte read and 128-byte write FIFOs providing a buffer from and to the processor.

This SPI controller can be used to communicate with a variety of peripherals such as memories, temperature sensors, pressure sensors, ADC, real-time clocks, LCD displays, and any SD card.

17.1.1 Features

The PS supports two SPI controller devices with these key features:

- Four wire bus – MOSI, MISO, SCK, SS
- Full-duplex operation offers simultaneous receive and transmit
- Master mode
 - Manual or auto start transmission of data
 - Manual or auto slave select (SS) mode
 - Supports up to three slave select lines
 - Allows the use of an external peripheral select 3-to-8 decode to increase the number of SS
 - Programmable delays for data transmission
- Slave mode
 - Programmable start detection mode
- Multi-master environment
 - Drives into 3-state if not enabled
 - Identifies an error condition if more than one master is detected
- Supports a 50 MHz maximum external SPI clock rate via the MIO
 - 25 MHz maximum via EMIO to PL pins
- Selectable master clock reference

- Programmable master baud rate divisor
- Supports 128-byte read and 128-byte write FIFOs
 - Each FIFO is 8-bits wide
- Programmable FIFO thresholds
- Supports programmable clock phase and polarity
- Supports manual or auto start transmission of data
- Software can poll for status or function as interrupt-driven device
- Programmable interrupt generation

17.1.2 System Viewpoint

The system viewpoint diagram of the SPI controller is shown in Figure 17-1.

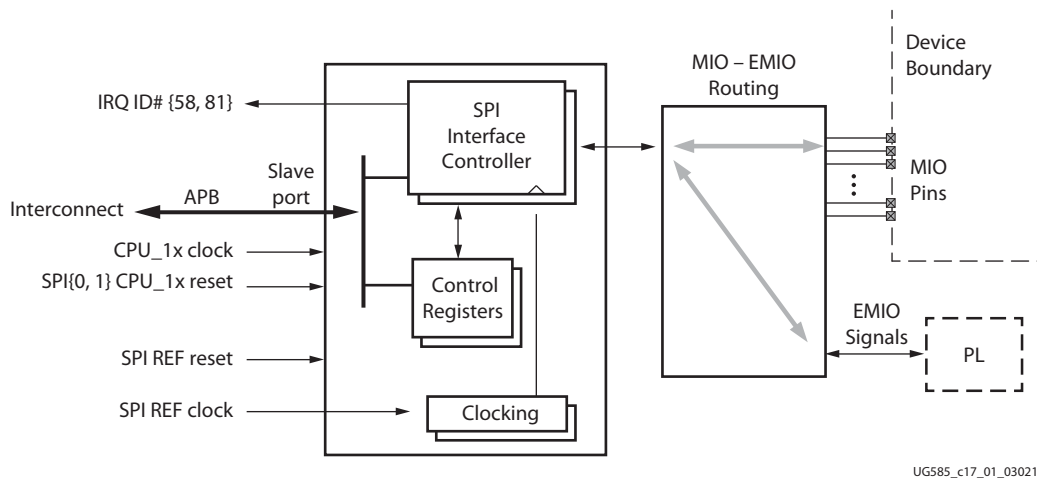


Figure 17-1: SPI System Block Diagram

17.2 Functional Description

17.2.1 Block Diagram

A functional block diagram of the SPI controller is shown in [Figure 17-2](#).

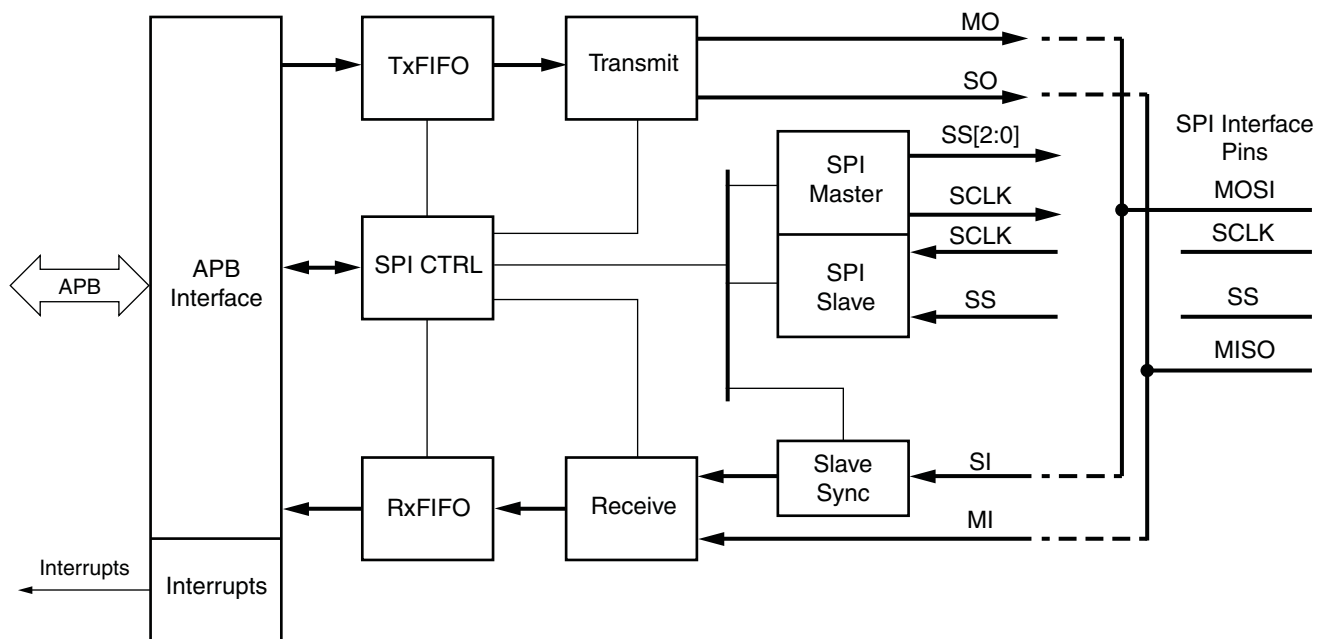


Figure 17-2: SPI Peripheral Block Diagram

17.2.2 Master Mode

In master mode, the SPI interface can transmit data to a slave or initiate a transfer to receive data from a slave. The controller can select only one slave device at the time using one of the three slave select lines (SS). If more than three slave devices need to be connected to the master, it is possible to add an external peripheral select 3-to-8 decode on the board. During the entire transmission the SS must remain active.

The master can operate in manual SS mode where the SS signals are toggled by writing to the register, or in auto SS mode where the SS signals are handled automatically by the controller. The auto mode is more efficient than manual SS mode. However, since the SPI controller is connected to a switch which may be busy with other operations, it might be difficult to guarantee that the command is coming in fast enough. In auto mode, if input commands cannot keep up, the FIFO would become empty, and SS would be deasserted. In this case, the manual start mode can be used to pre-fill the FIFO before starting the write operation. A threshold interrupt can then be used to fill further data into the FIFO, before it becomes empty. Manual SS mode is fully under software control which can guarantee proper SS behavior independently from the data flow.

Data to be transmitted is loaded into the TxFIFO and then unloaded for transmission onto the master output (MOSI) pin. Transmission is continuous while there is still data in the TxFIFO. By default, transmission starts as soon as there is any data in the TxFIFO.

Manual start mode can be enabled through bit 15 of the Configuration register. This allows the TxFIFO to be filled before starting the write operation. The write operation is then started by writing a start command to bit 16 in the Configuration register.

Data received from the slave through the master input (MISO) is loaded into the RX Data register serially. After a complete data word has been received, it is loaded into the RxFIFO. The APB can then read the word from the RxFIFO.

When the SPI controller is configured to be a master and it is not enabled, it is driven into a 3-state condition to allow other masters on the link to drive data onto the select lines. It also can detect errors associated with multi-master operation.

The value of the clock phase bit, *cpha*, is configured via bit 2 of the Configuration register. It has the following effects when the SPI controller is in master mode:

- When *cpha* = 0, the master SPI automatically toggles the slave chip select outputs inactive High for a minimum of one *spi_ref_clk* (or one *ext_clk* cycle if *ext_clk* is configured) between word transfers (configurable via the Delay register).
- When *cpha* = 1, the chip select outputs are not switched inactive between words.
- When *cpha* = 0, the minimum delay between the last bit period of the current word and the first bit period on the next word is three *spi_ref_clk* cycles or three *ext_clk* cycles (configurable via the Delay register). This enables the TxFIFO to be unloaded and ready for the next parallel-to-serial conversion and to toggle chip select inactive High.
- When *cpha* = 1, the minimum delay between the last bit period of the current word and the first bit period on the next word is, by default, one *spi_ref_clk* cycles or one *ext_clk* cycle (configurable by the Delay register). This allows the TxFIFO to be unloaded and ready for the next parallel-to serial-conversion.

Note: When the FIFO goes empty, the slave chip select outputs go inactive. Additionally, if manual start was used, it will need to be restarted manually.

17.2.3 Multi-Master Support

The SPI has a concept of multi-master mode and can detect errors associated with multi-master operation. When the SPI controller is configured to be a master and is not enabled, it is driven into a 3-state condition to allow other masters on the link to drive data onto the select lines. If it is driven active while the SPI is configured as a master, it is assumed that there has been multi-master bus contention. The output drivers of the SPI pins are switched off immediately by resetting the SPI enable bit. Additionally, the Interrupt Status register indicates a mode fault.

17.2.4 Slave Mode

In slave mode, the SPI module receives messages from an external SPI master and outputs a simultaneous reply. To synchronize the SLAVE to the external MASTER, it is vital that the SLAVE logic detect SPI word boundaries. Synchronization can be achieved when either of these start conditions is detected:

- If the slave select (SS) port is High (inactive), the next active edge on SCLK after SS transitions to Low (active) is considered as the start of a word.
- If a SPI slave is enabled when slave select (SS) was set Low (active), the slave controller detects the next word boundary by counting the number of reference clock cycles that SCLK is in an inactive state. If this number matches the number stored in the slave idle count register, synchronization has been achieved.
 - For example, if the sclk_in was a quarter the frequency of spi_ref_clk, then setting the slave_idle_count register to 2 would be sufficient.

Note: The start condition must be held for at least four spi_ref_clk cycles to be detected.

If the slave is enabled at a time when the external master is very close to starting data transfer, there is a small probability that false synchronization will occur, causing packet corruption. This issue can be avoided by any of the following means:

- Ensuring that the external master does not initiate data transfer until at least 10 spi_ref_clk cycles have passed after the slave has been enabled.
- Always ensure that the slave is enabled before the external master is enabled.
- Ensure that n_ss_in is NOT active when the slave is enabled.

17.2.5 SPI FIFOs

The FIFOs are asynchronous. The TxFIFO is written in the APB clock domain and read by a reference clock, decoupling the APB clock domain from the SPI clock domain. The RxFIFO is written using the reference clock and read in the APB clock domain.

When the RX FIFO is full and an attempt is made to push data into the RxFIFO then the overflow flag is set. This flag remains set until the RxFIFO is read and the flag is then deactivated. No data is added to the RxFIFO when it is full. Thus, all data received by the slave while the RxFIFO is full is lost. FIFO overflows or underflows can only occur when the SPI is configured in SLAVE mode.

Thresholds for both FIFOs are also provided to enhance the control on the data flow.

See SPI register map in [Appendix B, Register Details](#) for details of register fields.

Note: Some of the FIFO level triggers will not be set immediately on the actual internal events, due to clock synchronization delays between spi_ref_clk and CPU_1x clocks. This delay depends on the difference in frequencies between spi_ref_clk and CPU_1x clocks; the larger the difference, the larger the delay. To avoid FIFO overflows and underflows, the user should set the FIFO threshold values appropriately.

17.2.6 SPI Clocks

The SPI clocks are generated by the SPI reference clock block described in [Chapter 25, Clocks](#).

SPI Clock Rules:

- The SPI reference clock (spi_ref_clk) must be set to a higher frequency than the CPU_1x clock frequency.
- When in slave mode, the sclk_in clock must be, at maximum, a divide by four of spi_ref_clk. It can be any frequency less than a divide by four of spi_ref_clk.
- When in master mode, the sclk_out clock must be at maximum, a divide by four of spi_ref_clk. It can be any frequency less than a divide by four of spi_ref_clk.
- The user must allow for sufficient time to keep both the TxFIFO and Rx FIFO from underflowing and overflowing, respectively. The rate at which data is transferred over the serial link is defined by the frequency of sclk_in, whereas words are transferred into and from the FIFOs in the CPU_1x clock domain, using the APB interface. This means there is a fundamental minimum frequency for the CPU_1x clock, and this is determined by the frequency of sclk_in.

Note: The user is responsible to ensure that the dividers are set appropriately and should not rely on default values

17.2.7 SPI EMIO Considerations

The SPI interfaces are enabled through the MIO as well as EMIO interfaces. They are mutually exclusive in that once the interface is routed through the MIO, it no longer is available via the EMIO. If the user chooses to use the EMIO interface for SPI due to other MIO priorities, the user should know that the maximum operating frequency for SPI via EMIO is limited. The PL connectivity should connect the PS SPI EMIO interface directly to PL pads.

17.3 I/O Interface Signals

SPI interface signals are identified in [Table 17-1](#). The MIO pins and any restrictions based on device version are shown in the MIO table in section [2.4.4 MIO-at-a-Glance Table](#).

Table 17-1: SPI MIO Pins and EMIO Signals

SPI Interface	Controller Default Input Value	MIO Pins		EMIO Signals	
		Numbers	I/O	Name	I/O
SPI 0 Clock	0	16, 28, 40	IO	EMIOSPI0SCLKI	I
				EMIOSPI0SCLKO	O
				EMIOSPI0SCLKTN	O
SPI 0 MOSI	0	21, 33, 45	IO	EMIOSPI0SI	I
				EMIOSPI0MO	O
				EMIOSPI0MOTN	O

Table 17-1: SPI MIO Pins and EMIO Signals (Cont'd)

SPI Interface	Controller Default Input Value	MIO Pins		EMIO Signals	
		Numbers	I/O	Name	I/O
SPI 0 MISO	0	17, 29, 41	IO	EMIOSPI0MI	I
				EMIOSPI0SO	O
				EMIOSPI0STN	O
SPI 0 Slave Select 0	1	18, 30, 42	IO	EMIOSPI0SSIN	I
				EMIOSPI0SSON0	O
SPI 0 Slave Select 1	~	19, 31, 43	O	EMIOSPI0SSON1	O
SPI 0 Slave Select 2	~	20, 32, 44	O	EMIOSPI0SSON2	O
SPI 0 SS Tristate	~	~	~	EMIOSPI0SSNTN	O
SPI 1 Clock	0	12, 24, 36, 48	IO	EMIOSPI1SCLKI	I
				EMIOSPI1SCLKO	O
				EMIOSPI1SCLKTN	O
SPI 1 MOSI	0	10, 22, 34, 46	IO	EMIOSPI1SI	I
				EMIOSPI1MO	O
				EMIOSPI1MOTN	O
SPI 1 MISO	0	11, 23, 35, 47	IO	EMIOSPI1MI	I
				EMIOSPI1SO	O
				EMIOSPI1STN	O
SPI 1 Slave Select 0	1	13, 25, 37, 49	IO	EMIOSPI1SSIN	I
				EMIOSPI1SSON0	O
SPI 1 Slave Select 1	~	14, 26, 38, 50	O	EMIOSPI1SSON1	O
SPI 1 Slave Select 2	~	15, 27, 39, 51	O	EMIOSPI1SSON2	O
SPI 1 SS Tristate	~	~	~	EMIOSPI1SSNTN	O

CAN Controller

18.1 Introduction

This chapter describes the architecture and features of the CAN controllers and the functions of the various registers in the design. There are two nearly identical CAN controllers in the PS that are independently operated. Defining the CAN protocol is outside the scope of this document, and knowledge of the specifications is assumed.

18.1.1 Features

Key features of the CAN Controller are summarized as follows:

- Compatible with the *ISO 11898 -1*, *CAN 2.0A*, and *CAN 2.0B* standards
- Standard (11-bit identifier) and extended (29-bit identifier) frames
- Bit rates up to 1 Mb/s
- Transmit message FIFO (TxFIFO) with a depth of 64 messages
- Transmit prioritization through one high-priority transmit buffer (TxHPB)
- Watermark interrupts for TxFIFO and RxFIFO
- Automatic re-transmission on errors or arbitration loss in normal mode
- Receive message FIFO (RxFIFO) with a depth of 64 messages
- Four Rx acceptance filters with enables, masks and IDs
- Loopback and snoop modes for diagnostic applications
- Maskable error and status interrupts
- 16-Bit time stamping for receive messages
- Readable Rx/Tx error counters

18.1.2 System Viewpoint

The system viewpoint of the CAN controller is shown in Figure 18-1.

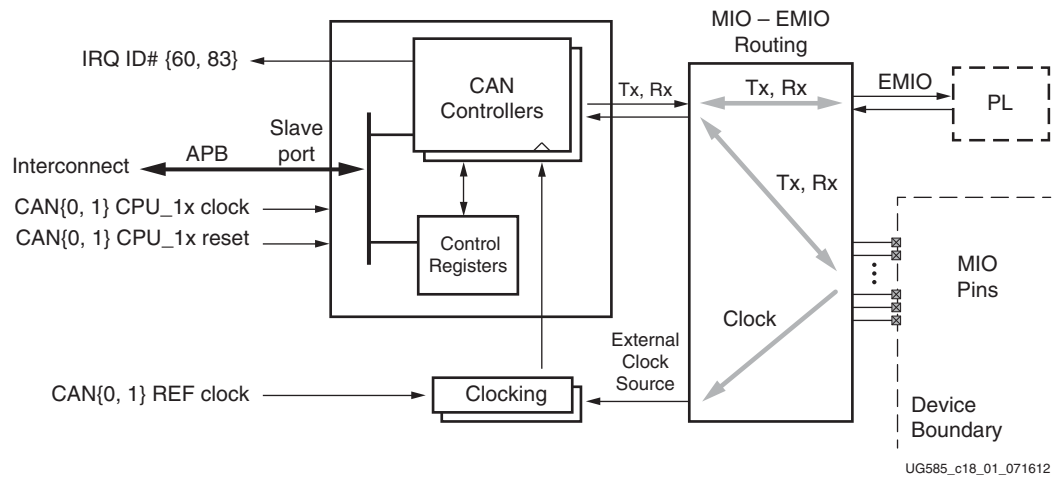


Figure 18-1: CAN Controller System Viewpoint

18.1.3 Block Diagram

The high-level architecture of the CAN core is shown in Figure 18-2. The sub-modules are described in subsequent sections.

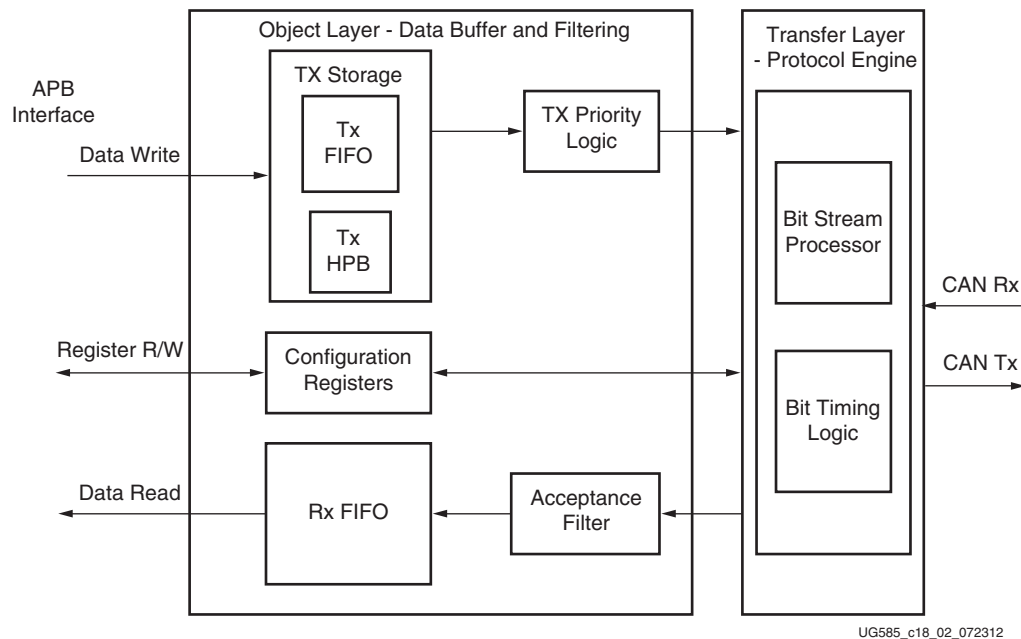


Figure 18-2: CAN Controller Block Diagram

Configuration Registers

The CAN Controller Configuration register defines the configuration registers. This module allows for read and write access to the registers through the APB interface. An overview of the CAN controller registers are shown in section [18.3.8 Register Overview](#).

Transmit and Receive Messages

Separate storage buffers exist for transmit (TxFIFO) and receive (Rx FIFO) messages through a FIFO structure. The depth of each buffer is a maximum of 64 messages.

Tx High Priority Buffer

Each controller also has a transfer high priority buffer (TxHPB) provides storage for one transmit message. Messages written on this buffer have maximum transmit priority. They are queued for transmission immediately after the current transmission is complete, preempting any message in the Tx FIFO.

Acceptance Filters

Acceptance filters sort incoming messages with the user-defined acceptance mask and ID registers to determine whether to store messages in the Rx FIFO, or to acknowledge and discard them. Messages passed through acceptance filters are stored in the Rx FIFO.

18.1.4 Notices

Restrictions

There is a single PS clock generator for both controllers. When the internal clock is used, it will be of the same clock frequency, but the clock to each controller can be individually enabled using the slcr register, see section [25.7.4 CAN Clocks](#). Also, either or both controllers can be clocked from an external source via an MIO pin, see section [18.4.1 Clocks](#).

18.2 Functional Description

Each controller is independently configured and controlled. There are two CAN controllers (CAN_x; where x = 0 or 1). The register name preface for the CAN registers is 'can' (e.g., can.MSR register).

18.2.1 Controller Modes

Mode Transitions

The supported mode transitions are shown in [Figure 18-3](#). The transitions are primarily controlled by the resets, the CEN bit, the MSR register settings, and a hardware wake up mechanism.

To enter normal mode from configuration mode:

- Clear can.MSR[LBACK, SNOOP, SLEEP] = 0
- Set can.SRR[CEN] = 1

To enter sleep mode from normal mode (interrupt generated):

- Set can.MSR[SLEEP] = 1

Events that cause the controller to exit sleep mode (interrupt generated):

- Rx signal activity (hardware sets can.MSR[SLEEP] = 0)
- TxFIFO or TxHPB activity (hardware sets can.MSR[SLEEP] = 0)
- Software writes 0 to can.MSR[SLEEP]

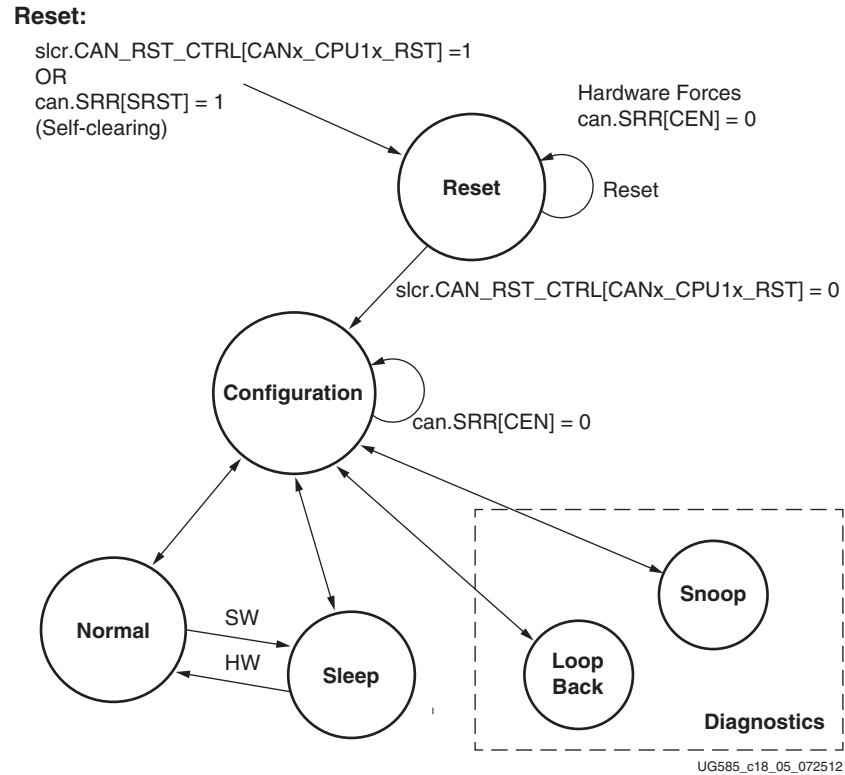


Figure 18-3: CAN Operating Mode Transitions

Mode Settings

Table 18-1 defines the CAN controller modes of operation and corresponding control and status bits.

Table 18-1: CAN Controller Modes of Operation

CAN CPU_1x PS Reset (slcr)	Software Reset Register (can.SRR)		Mode Select Register (MSR) (Read/Write bits)			Status Register (SR) (Read Only bits)					Operational Mode
	SRST (CAN Reset)	CEN (CAN Enable)	LBACK	SLEEP	SNOOP	CONFIG	LBACK	SLEEP	NORMAL	SNOOP	
1	X	X	X	X	X	1	0	0	0	0	Reset
0	1	X	X	X	X	1	0	0	0	0	Reset
0	0	0	X	X	X	1	0	0	0	0	Configuration
0	0	1	1	X	X	0	1	0	0	0	Loop back
0	0	1	0	1	0	0	0	1	0	0	Sleep
0	0	1	0	0	1	0	0	0	1	1	Snoop
0	0	1	0	0	0	0	0	0	1	0	Normal

Normal Mode

Normal mode transmits and receives messages on the Tx and Rx I/O signals as defined by the Bosch and IEEE specifications.

Sleep Mode

Sleep mode can be used to save a small amount of power during idle times. When in sleep mode, the controller can transition to normal mode or configuration mode. In sleep mode:

- When another node transmits a message, the controller receives the message and exits sleep mode.
- If there is a new Tx request then the hardware switches the controller to normal mode and the controller services the request(s).
- Interrupt can be generated when the controller enters sleep mode.
- Interrupt can be generated when the controller wakes up.

Sleep mode is exited by the hardware when there is CAN bus activity or a request in either TxFIFO or TxHPB. When the controller exits sleep mode, `can.MSR[SLEEP]` is set to 0 by the hardware and an interrupt can be generated.

Loop Back Mode (Diagnostics)

Loop back mode is used for diagnostic purposes. When in loop back mode, the controller must only be programmed to enter configuration mode or be held in reset. In loop back mode:

- Controller transmits a recessive bitstream onto the CAN_TX bus signal.
- Tx messages are internally looped back to the Rx line and are acknowledged.
- Tx messages are not sent on the CAN_TX bus signal.
- The controller receives all message that it transmits.
- The controller does not receive any messages transmitted by other CAN nodes.

Snoop Mode (Diagnostics)

Snoop mode is used for diagnostic purposes. When in snoop mode, the controller must only be programmed to enter configuration mode or be held in reset. In snoop mode:

- The controller transmits a recessive bitstream onto the CAN bus.
- The controller does not participate in normal bus communication.
- The controller receives messages that are transmitted by other CAN nodes.
- Software can program acceptance filters to dynamically enable/disable and change criteria.

Error counters are disabled and cleared to 0. Reads to error counter registers return 0.

18.2.2 Message Format

The same message format is used for RxFIFO and Tx (FIFO and HPB). Each message includes four words (16 bytes). Software must read and write all four words regardless of the actual number of data bytes and valid fields in the message.

The message words, fields and structure are shown in [Table 18-2](#). The details of each field are in [Table 18-3](#).

Table 18-2: CAN Message Format

	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
Message Identifier [IDR]	ID[28:18]											STR/RTR	IDE	ID[17:0]																	RTR	
Data Length Code [DLCR]	DLC[3:0]			Reserved											Time Stamp (Rx only, Reserved for Tx)																	
Data Word 1 [DW1R]	DB0[7:0]					DB1[7:0]					DB2[7:0]					DB3[7:0]																
Data Word 2 [DW2R]	DB4[7:0]					DB5[7:0]					DB6[7:0]					DB7[7:0]																

Bit Field Details

Writes

If a bit field or data byte is not required, then write zeros. Software should write the default values shown in [Table 18-3](#) for unused functions.

Reads

Data starts at byte 0 and continues for the number of counts in DLC. Software should read both data words, but the only valid bytes are determined by DLC.

[Table 18-3](#) provides bit descriptions for the identifier word bits, the DLC word bits, and data word 1 and data word 2 bits.

Table 18-3: CAN Message Word Register Bit Fields

Bits	Name	Default Value	Description
Identifier Word			
31:21	ID[28:18]	0	Standard Message ID The identifier portion for a standard frame is 11 bits. These bits indicate the standard frame ID. This field is valid for both standard and extended frames.
20	SRR/RTR	0	Substitute Remote Transmission Request This bit differentiates between data frames and remote frames. Valid only for standard frames. For extended frames this bit is 1. 1: Indicates that the message frame is a Remote Frame. 0: Indicates that the message frame is a Data Frame.
19	IDE	0	Identifier Extension This bit differentiates between frames using the Standard Identifier and those using the Extended Identifier. Valid for both Standard and Extended Frames. 1: Indicates the use of an extended message identifier. 0: Indicates the use of a standard message identifier.
18:1	ID[17:0]	0	Extended Message ID This field indicates the extended identifier. Valid only for extended frames. For standard frames, reads from this field return 0s. For Standard frames, writes to this field should be 0s.
0	RTR	0	Remote Transmission Request This bit differentiates between data frames and remote frames. Valid only for extended frames. 1: Indicates the message object is a remote frame. 0: Indicates the message object is a data frame. For standard frames, reads from this bit returns 0. For standard frames, writes to this bit should be 0.
DLC Word			
31-28	DLC	0	Data Length Code This is the data length portion of the control field of the CAN frame. This indicates the number valid data bytes (0 to 8) that are in the Data Word 1 and Data Word 2 registers.
27-0	Reserved	0	Reads from this field return 0s. Writes to this field should be 0s.
Data Word 1			
DW1R[31:24]	DB0[7:0]	0	Data Byte 0
DW1R[23:16]	DB1[7:0]	0	Data Byte 1
DW1R[15:8]	DB2[7:0]	0	Data Byte 2
DW1R[7:0]	DB3[7:0]	0	Data Byte 3

Table 18-3: CAN Message Word Register Bit Fields (Cont'd)

Bits	Name	Default Value	Description
Data Word 2			
DW2R[31:24]	DB4[7:0]	0	Data Byte 4
DW2R[23:16]	DB5[7:0]	0	Data Byte 5
DW2R[15:8]	D6[7:0]	0	Data Byte 6
DW2R[7:0]	DB7[7:0]	0	Data Byte 7

18.2.3 Message Buffering

Rx Messages

The RxFIFO can store up to 64 Rx CAN messages that are received and optionally filtered. Rx messages that pass any of the acceptance filters are stored in the RxFIFO. When no acceptance filter has been selected, all received messages are stored in the RxFIFO. Software reads these messages as described in [18.3.7 Read Messages from RxFIFO](#).

A timestamp is added to each successfully stored Rx message. A free running 16-bit counter is clocked using the CAN bit time clock. The rules for time stamping an Rx message are:

- The counter rolls over. There is no status bit to indicate that the roll over condition occurred.
- The timestamp included when a Rx message is successfully collected. The sampling of the counter takes place at the last bit of EOF.
- The counter is cleared when CEN=0 or by software writing a 1 to the can.TCR register.

Software must read all four registers of an Rx message in the RxFIFO, regardless of how many data bytes are in the message. The first word is read using the RXFIFO_ID register and contains the identifier of the received message (IDR). The second word is read using the RXFIFO_DLC register and contains the 16-bit timestamp and data length code (DLC) field. The third and fourth words contain data word 1 (DW1R) and data word 2 (DW2R).

Writes to the RxFIFO registers are ignored. Read data from an empty RxFIFO are invalid and may generate an interrupt.

The messages in the RxFIFO are retained even if the CAN controller enters Bus off state or Configuration mode.

Tx Messages

The controller has a configurable TxFIFO that software can use buffer up to 64 Tx CAN messages. The controller also has a high priority transmit buffer (Tx HPB), with storage for one message. When a higher priority message needs to be sent, software writes the message to the high priority transmit buffer when it is available. The message in the TxHPB has higher priority over messages in the TxFIFO.

When arbitration loss or errors occur during the transmission of a message, the controller tries to retransmit the message. No subsequent message, even a newer, higher priority message is transmitted until the original message is transmitted without errors or arbitration loss.

The controller transmits the message starting with bit 31 of the IDR word, see table 18-2, CAN Message Format. After the identifier word is transmitted, the DLCR word is transmitted. This is followed by the data bytes in this order: DB0, DB1, ... DB7. The last bit in the data portion of the message is DB7, bit 0. Refer to [Table 18-2](#).

The status bit, can.ISR[TXOK] is set = 1 after the controller successfully transmits a message from either the Tx FIFO or TxHPB.

The messages in the Tx FIFO and TxHPB are retained even if the CAN controller enters Bus off state or Configuration mode.

The message format is described in [18.2.2 Message Format](#).

Reads from Rx FIFO

The byte stream of status and data received from the flash device is stored in the Rx FIFO. The controller has a single read-only 32-bit RXD register for software to read the byte stream received from the flash device. Software checks the interrupt status registers for the read data bytes presence in the Rx FIFO. Software should only read the RXD register when there is data present in Rx FIFO. Please note that reading RXD register when the Rx FIFO is empty results in setting of the Rx FIFO underflow interrupt status bit.

All 16 bytes must be read from the Rx FIFO to receive the complete message. The first word read (4 bytes) returns the identifier of the received message (IDR). The second read returns the 16-bit receive time stamp and data length code (DLC) field of the received message (DLCR). The third read returns data word 1 (DW1R), and the fourth read returns data word 2 (DW2R).

A free running 16-bit counter provides a time stamp relative to the time the message was successfully received.

All four words must be read for each message, even if the message contains less than eight data bytes. Write transactions to the Rx FIFO are ignored. Reads from an empty Rx FIFO return invalid data.

Rx and Tx Error Counters

When an Rx or Tx error occurs, the associated error counters in the protocol engine (see section [18.2.6 Protocol Engine](#)) are incremented. The two error counters are 8 bits wide and are read using the read-only can.ECR register, bit fields REC and TEC.

The Rx and Tx counters are reset when any the these situations occur:

- After a 1 is written to can.SRR[SRST] field = 1. This bit write is self-clearing.
- Anytime can.SRR[CEN] = 0 (configuration mode).
- When the controller enters Bus Off state.

18.2.4 Interrupts

Each CAN controller has a single interrupt signal to the GIC interrupt controller. CAN 0 connects to IRQ ID#60 and CAN 1 connects to ID #83. The source of an interrupt can be grouped into one of the following:

- TxFIFO and TxHPB
- RxFIFO
- Message passing and arbitration
- Sleep mode and bus off

Enable and disable interrupts using the can.IER register. Check the raw status of the interrupt using can.ISR. Clear interrupts by writing a 1 to can.ICR. Some interrupt sources have an additional method to clear the interrupt as shown in [Table 18-4](#).

List of Interrupts

All of the CAN interrupts are sticky; that is, once the hardware sets them they stay set until cleared by software. CAN status and interrupts are identified in [Table 18-4](#).

Table 18-4: List of CAN Status and Interrupts

Name	Bit Number	Additional Method to Clear Interrupt	Usage
TxFIFO Watermark	13	none	Operational threshold.
TxFIFO Empty	14	none	Empty indicator.
TxFIFO Full	2	none	Full indicator.
TxHPB Full	3	none	This status indicates if the buffer is in-use and should not be written.
RxFIFO Watermark	12	none	Operational threshold.
RxFIFO Not Empty	7	none	One or more messages can be read.
RxFIFO overflow	6	Write 0 to can.SRR[CEN]	FIFO was full and message(s) likely lost.
RxFIFO underflow	5	none	Programming error, message read from RxFIFO when no messages were there.
Message Rx	4	Write 0 to can.SRR[CEN]	
Message Tx	1	Write 0 to can.SRR[CEN]	
Message Error	8	Write 0 to can.SRR[CEN]	Any of the five CAN errors occurs.
Arbitration Lost	0	none	
Enter Sleep Mode	10	Write 0 to can.SRR[CEN]	

Table 18-4: List of CAN Status and Interrupts

Name	Bit Number	Additional Method to Clear Interrupt	Usage
Exit Sleep Mode	11	Write 0 to can.SRR[CEN]	Controller can go to normal or configuration mode.
Bus Off	9	Write 0 to can.SRR[CEN]	

RxFIFO and TxFIFO Interrupts

The FIFO watermark levels and all the FIFO interrupts are illustrated in Figure 18-4, CAN RxFIFO and TxFIFO Watermark Interrupts.

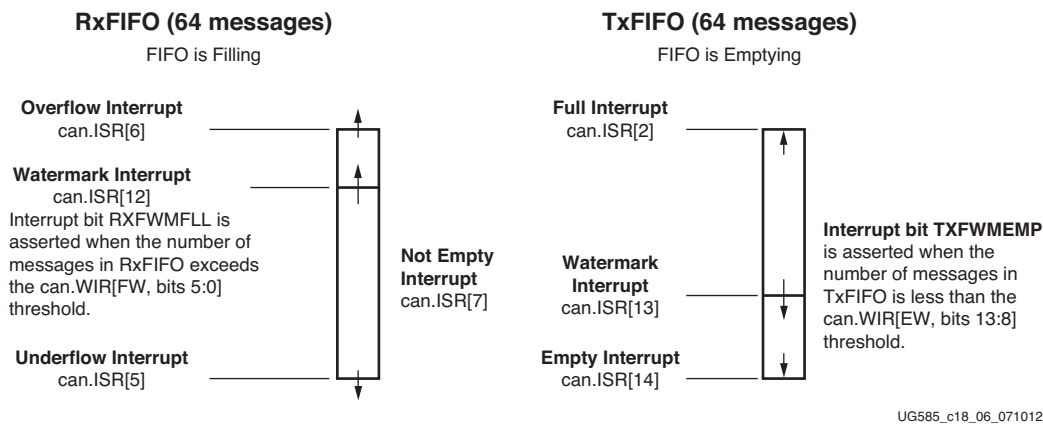


Figure 18-4: CAN RxFIFO and TxTxFIFO Watermark Interrupts

Example: Program RxFIFO Watermark Interrupt (12)

The following steps can be used to setup and control the RxFIFO watermark interrupt. See Figure 18-4. The watermark status and control interrupts are described in Protocol Engine, page 438.

1. **Disable RxFIFO watermark interrupt.** Write a 0 to can.IER[12].
2. **Program RxFIFO full watermark level.** Write to can.WIR[FW].
3. **Clear RxFIFO watermark interrupt.** Write a 1 to can.ICR[12].
4. **Read RxFIFO watermark status.** Read can.ISR[12].
5. **Enable RxFIFO watermark interrupt.** Write a 1 to can.IER[12].

Example: Program TxTxFIFO Watermark Interrupt (13)

The following steps can be used to setup and control the TxTxFIFO watermark interrupt. See Figure 18-4. The watermark status and control interrupts are described in Protocol Engine, page 438.

1. **Disable TxTxFIFO watermark interrupt.** Write a 0 to can.IER[13].
2. **Program TxTxFIFO empty watermark level.** Write to can.WIR[EW].

3. **Clear TxFIFO watermark interrupt.** Write a 1 to can.ICR[13].
4. **Read TxFIFO watermark status.** Read can.ISR[13].
5. **Enable TxFIFO watermark interrupt.** Write a 1 to can.IER[13].

Example: Program TxFIFO Empty Interrupt (14)

The following steps can be used to control the TxFIFO empty interrupt:

1. **Disable TxFIFO empty interrupt.** Write a 1 to can.IER[14].
2. **Clear TxFIFO empty interrupt.** Write a 1 to can.ICR[14].
3. **Enable TxFIFO empty interrupt.** Write a 1 to can.IER[14].

Read TxFIFO empty status. Read can.ISR[14]. It indicates the status whether TxFIFO is empty or not.

18.2.5 Rx Message Filtering

To filter Rx messages, configure and enable up to four acceptance filters with acceptance mask and ID registers to determine whether to store messages in the RxFIFO, or to acknowledge and discard them.

Acceptance filtering is performed in the following sequence:

1. The incoming identifier is masked with the bits in the Acceptance Filter Mask register.
2. The Acceptance Filter ID register is also masked with the bits in the Acceptance Filter Mask register.
3. Both resulting values are compared.
4. If both these values are equal, then the message is stored in the RxFIFO.
5. Acceptance filtering is processed by each of the defined filters. If the incoming identifier passes through any acceptance filter, then the message is stored in the RxFIFO.

Acceptance Filter Enable

The Acceptance Filter register (AFR) defines which acceptance filters to use. It includes four enable bits that correspond to the four acceptance filters. Each Acceptance Filter ID register (AFIR) and Acceptance Filter Mask register (AFMR) pair is associated with a UAF bit.

When the UAF bit is 1, the corresponding acceptance filter pair is used for acceptance filtering. When the UAF bit is 0, the corresponding acceptance filter pair is not used for acceptance filtering.

To modify an acceptance filter pair in normal mode, the corresponding UAF bit in this register must first be set to 0. After the acceptance filter is modified, the corresponding UAF bit must be set to 1 for the filter to be enabled.

The UAF bits in the can.AFR register enable the Rx acceptance filters:

- If all UAF bits are set to 0, then all received messages are stored in the RxFIFO.
- If the UAF bits are changed from a 1 to 0 during reception of a CAN message, the message will not be stored in the RxFIFO.

If any of the enabled filters (up to four) satisfy this equation, then the Rx message is stored in the RxFIFO:

If (AFMR & Message_ID) == (AFMR & AFIR) then Capture Message

Each acceptance filter is independently enabled. The filters are selected by the can.AFR register.

- Set can.AFR[UAF4] = 1 to enable AFMR4 and AFID4.
- Set can.AFR[UAF3] = 1 to enable AFMR3 and AFID3.
- Set can.AFR[UAF2] = 1 to enable AFMR2 and AFID2.
- Set can.AFR[UAF1] = 1 to enable AFMR1 and AFID1.

If all can.AFR[UAFx] bits are set = 0, then all received messages are stored in the RxFIFO. The UAF bits are sampled by the hardware at the start of an incoming message.

Acceptance Filter Mask

The Acceptance Filter Mask registers (AFMR) contain mask bits used for acceptance filtering. The incoming message identifier portion of a message frame is compared with the message identifier stored in the Acceptance Filter ID register. The mask bits define which identifier bits stored in the Acceptance Filter ID register are compared to the incoming message identifier.

There are four AFMRs. These registers are stored in a memory. Reads from AFMRs return 'X's if the memory is uninitialized. Asserting a software reset or hardware reset does not clear register contents. These registers can be read from and written to. These registers are written to only when the corresponding UAF bits in the can.AFR register are 0 and the ACFBSY bit in the can.SR register is 0.

The following conditions govern AFMRs:

Extended Frames All bit fields (AMID [28:18], AMSRR, AMIDE, AMID [17:0] and AMRTR) need to be defined.

Standard Frames Only AMID [28:18], AMSRR and AMIDE need to be defined. AMID [17:0] and AMRTR should be written as 0.

Acceptance Filter Identifier

The Acceptance Filter ID registers (AFIR) contain Identifier bits, which are used for acceptance filtering. There are four Acceptance Filter ID registers.

These registers can be read from and written to. These registers should be written to only when the corresponding UAF bits in the SR are 0 and the ACFBSY bit in the SR is 0.

The following conditions govern the use of the AFIRs:

Extended Frames All the bit fields (AIID [28:18], AISRR, AIIDE, AIID [17:0] and AIRTR) must be defined.

Standard Frames Only AIID [28:18], AISRR and AIIDE need to be defined. AIID [17:0] and AIRTR should be written with 0.

The user must ensure proper programming of the IDE bit for standard and extended frames. If the user sets the IDE bit in AMIR to 0, then it is considered to be a standard frame ID check only.

Example: Program Acceptance Filter

Each acceptance filter has its own mask, can.AFMR{1,2,3,4}, and ID register, can.AFIR{1,2,3,4}.

1. **Disable acceptance filters.** Write 0 to the can.AFR register.
2. **Wait for filter to be not busy.** Read can.SR[ACFBSY] until = 0.
3. **Write a filter mask and ID.** Write to a pair of AFMR and AFIR registers (refer to the example below).
4. **Write additional filter masks and IDs.** Go to step 2.
5. **Enable one or more Filters.** To enable all filters, write 0x0000_000F to the can.AFR register.

Program the AFMR and AFIR Registers

The valid fields for sending Tx messages to the controller are summarized in [Table 18-5](#). These fields are described in section [18.2.2 Message Format](#).

Table 18-5: CAN Message Identifier Register (IDR) Fields

	ID[28:18]	STR/RTR	IDE	ID[17:0]	RTR
Standard Frame	Valid	Valid	Valid	Ignored	Ignored
Extended Frame	Valid	Valid	Valid	Valid	Valid

In the AFMR mask register, enable (unmask) the compare functions for each field for the in-coming Tx CAN message by writing a 1 to the bit field. In the AFIR register, write the values that are to be compared to the in-coming Tx CAN message.

Example: Program the AFMR and AFIR for Standard Frames

This example setups up the acceptance filter for standard frames. The frame ID number is shown to be 0x5DF, but could be set to desired value for the application.

1. **Configure filter mask for standard frames.** Write 0xFFF8_0000 to the can.AFMR register:
 - a. Enable the compare for the standard message ID, [AMIDE] = 1.
 - b. Compare all bits in the standard message ID, [AMIDH] = 0x7FF.
 - c. Enable the compare for substitute remote transmission request, [AMSRR] = 1.
 - d. Zero-out the extended frame bits, [AMIDL, AMRTR] = 0.
2. **Configure filter ID for standard frames.** Write 0xABC0_0000 to the can.AFIR register:
 - a. Select the standard frame message mode, [AIIDE] = 0.
 - b. Program the standard message ID, [AIIDH] = 0x5DF (equals the value of 0xABC shifted right one bit).
 - c. Disable substitute remote transmission request, [AISRR] = 0.

- d. Zero-out extended frame bits, [AIIDL, AIRTR] = 0.

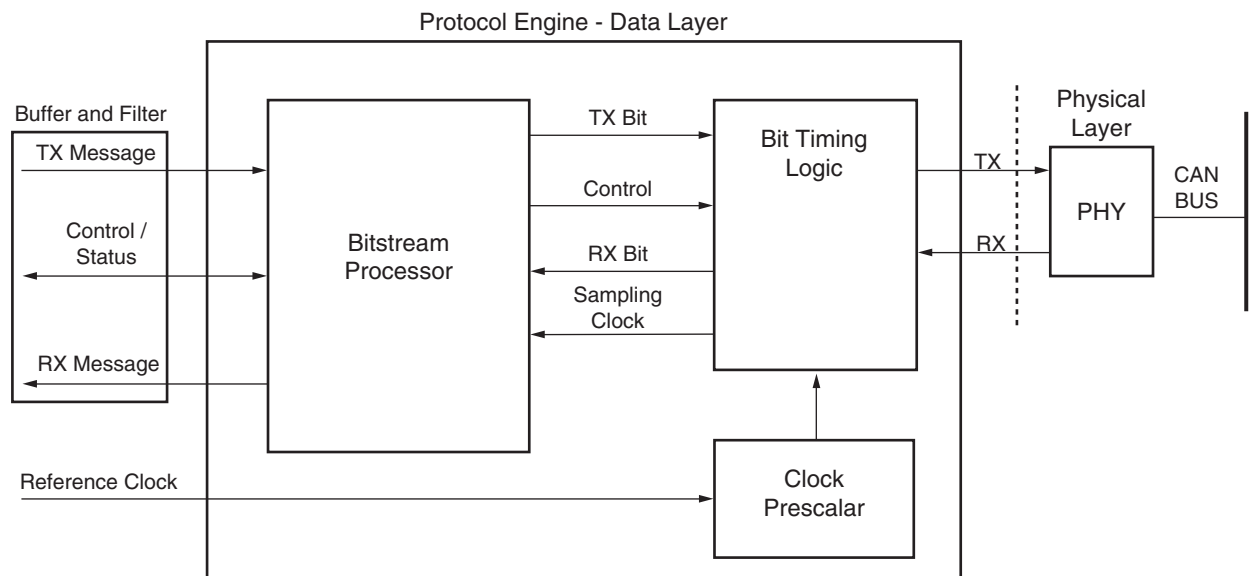
Example: Program the AFMR and AFIR for Extended Frames

This example setups up the acceptance filter for extended frames. The Frame ID number is shown to be 0x5DF, but could be set to desired value for the application.

1. **Configure filter mask for extended frames.** Write 0xFFFF_FFFF to the can.AFMR register:
 - a. Enable the substitute remote transmission request mask for frame, [AMSRR] = 1.
 - b. Compare all bits in the compare for the standard message ID, [AMIDH] = 0x7FF.
 - c. Enable the extended frame.
 - d. Extended ID.
 - e. Remote transmission request bit for extended frame.
2. **Configure filter ID for extended frames.** Write 0xABD1_7BDE to the can.AFIR register:
 - a. Standard ID, [AIIDH] = 0x5DF (equals the value of 0xABC shifted right one bit).
 - b. Remote transmission request bit for standard frame [AIRSRR] = 1.
 - c. Select standard/extended frame [AIIDE] = 1.
 - d. Extended ID [AIIDL] = 3DEF.
 - e. Remote transmission request bit for extended frame [AIRTR] = 0

18.2.6 Protocol Engine

The CAN protocol engine consists primarily of the bit timing logic (BTL) and the bitstream processor (BSP) modules. Figure 18-5 shows a block diagram of the CAN protocol engine.



UG585_c18_03_070912

Figure 18-5: CAN Protocol Engine

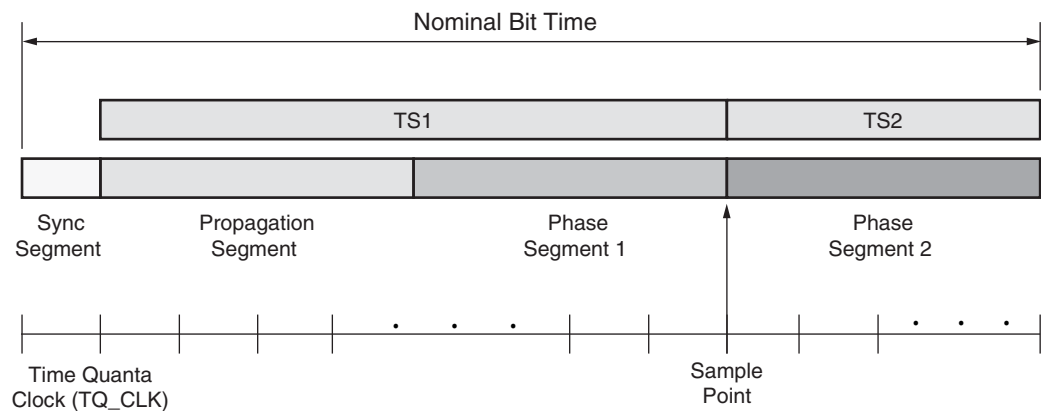
Rx/Tx Bit Timing Logic

The primary functions of the bit timing logic (BTL) module include:

- Generate the Rx sampling clock for the bitstream processor (BSP)
- Synchronize the CAN controller to CAN traffic on the bus
- Sample the bus and extracting the data stream from the bus during reception
- Insert the transmit bit stream onto the bus during transmission

The nominal length of the bit time clock period is based on the CAN_REF_CLK clock frequency, the baud rate generator divider (can.BRPR register) and the segment lengths (can.BTR register).

The bit timing logic module manages the re-synchronization function for CAN using the sync width parameter in the can.BTR[SJW] bit field. The CAN bit timing is shown in Figure 18-6.



UG585_c18_04_073012

Figure 18-6: CAN Bit Time

Sync Segment count always equals one time quanta period.

The TS 1 and TS 2 period counts are programmable using the can.BTR[TS1, TS2] bit fields. These registers are written when the controller is in configuration mode. The width of the propagation segment (PROP_SEG) must be less than the actual propagation delay.

Time Quanta Clock

The time quanta clock (TQ_CLK) is derived from the controller reference clock (CAN_REF_CLK) divided by the baud rate prescaler (BRP).

$$tTQ_CLK = tCAN_REF_CLK * (can.BRPR[BRP] + 1)$$

$$freqTQ_CLK = freqCAN_REF_CLK / (can.BRPR[BRP] + 1)$$

$$tSYNC_SEGMENT = 1$$

$$tTIME_SEGMENT1 = tTQ_CLK * (can.BPR[TS1] + 1)$$

$$tTIME_SEGMENT2 = tTQ_CLK * (can.BPR[TS2] + 1)$$

$$t_{\text{BIT_RATE}} = t_{\text{SYNC_SEGMENT}} + t_{\text{TIME_SEGMENT1}} + t_{\text{TIME_SEGMENT2}}$$

$$\text{freqBIT_RATE} = \text{freqCAN_REF_CLK} / (3 + \text{can.BPR[TS1]} + \text{can.BPR[TS2]})$$

Note: A given bit-rate can be achieved with several bit-time configurations, but values should be selected after careful consideration of oscillator tolerances and CAN propagation delays. For more information on CAN bit-time register settings, refer to the *CAN 2.0A*, *CAN 2.0B*, and *ISO 11898-1* specifications.

Bitstream Processor

The bitstream processor (BSP) module performs several functions while sending and receiving CAN messages. The BSP obtains a message for transmission from either the TxFIFO or the TxHPB and performs the following functions before passing the bitstream to the BTL.

- Serializing the message
- Inserting stuff bits, CRC bits, and other protocol defined fields during transmission

During transmission the BSP simultaneously monitors Rx data and performs bus arbitration tasks. It then transmits the complete frame when arbitration is won, and retrying when arbitration is lost.

During reception the BSP removes stuff bits, CRC bits, and other protocol fields from the received bitstream. The BSP state machine also analyses bus traffic during transmission and reception for Form, CRC, ACK, Stuff, and Bit violations. The state machine then performs error signaling and error confinement tasks. The CAN controller does not voluntarily generate overload frames but does respond to overload flags detected on the bus.

This module determines the error state of the CAN controller: error active, error passive or bus-off. When Tx or Rx errors are observed on the bus, the BSP updates the transmit and receive error counters according to the rules defined in the *CAN 2.0 A*, *CAN 2.0 B* and *ISO 11898-1* standards. Based on the values of these counters, the error state of the CAN controller is updated by the BSP.

18.3 Programming Guide

18.3.1 Overview

The controller has several operating modes and different ways to receive and transmit messages. The low-level functions were described in [18.2 Functional Description](#). The system-level operations are described in section [18.4 System Functions](#).

All the controller registers are listed in [Table 18-6](#) and are described in detail in [Appendix B, Register Details](#).

18.3.2 Configuration Mode State

The CAN controller enters configuration mode, irrespective of the operation mode, when any of these actions are performed:

- Writing a 0 to the CEN bit in the SRR register.
- Writing a 1 to the SRST bit in the SRR register. The controller enters Configuration mode immediately following the software reset.
- Driving a 1 on the Reset input controlled via SLCR. The controller continues to be in reset as long as Reset is 1. The controller enters configuration mode after Reset is negated to 0.

In configuration mode the following apply:

- The CAN controller loses synchronization with the CAN bus and drives a constant recessive bit on the bus line.
- The Error Count register (ECR) is reset.
- The Error Status register (ESR) is reset.
- The Bit Timing register (BTR) and Baud Rate Prescaler register (BRPR) can be modified.
- The CAN controller does not receive any new messages.
- The CAN controller does not transmit any messages. Messages in the TxFIFO and the TxHPB are appended. These packets are sent when normal operation is resumed.
- Reads from the RxFIFO can be performed.
- Writes to the TxFIFO and TxHPB can be performed (provided the SNOOP bit is not set).
- Interrupt Status register bits ARBLST, TXOK, RXOK, RXOFLW, ERROR, BSOFF, SLP, and WKUP are be cleared.
- Interrupt Status register bits RXNEMP and RXUFLW can be set due to read operations to the RxFIFO.
- Interrupt Status register bits TXBFLL and TXFLL, and Status register bits TXBFLL and TXFLL can be set due to write operations to the TX HPB and TX FIFO, respectively.
- Interrupts are generated if the corresponding bits in the Interrupt Enable register (IER) are 1.
- All Configuration registers are accessible.

When in configuration mode, the CAN controller stays in this mode until the CEN bit in the SRR register is set to 1. After the CEN bit is set to 1 the CAN controller waits for a sequence of 11 recessive bits before exiting configuration mode.

The CAN controller enters normal, loop back, snoop, or sleep modes from configuration mode, depending on the LBACK, SNOOP, and SLEEP bits in the MSR register.

18.3.3 Start-up Controller

The controller can operate in Normal, Sleep, Snoop and Loop Back modes. Refer to [Figure 18-3](#) for supported transitions. On start-up the controller clocks and configuration bits are programmed. Then the operating mode is selected and enabled.

Example: Start-up Sequence

1. **Configure clocks.** Refer to section [18.4.1 Clocks](#).
2. **Configure Tx/Rx signals.** Refer to section [18.5.1 MIO Programming](#).

3. Wait for configuration mode. Read can.SR[CONFIG] until it equals 1.
4. **Reset the controller.** The controller comes up in Configuration mode. Refer to section [18.4.2 Resets](#).
5. **Program the bit sampling clock.** Refer to section [Rx/Tx Bit Timing Logic](#).
6. **Program the interrupts, as needed.** Refer to section [18.2.4 Interrupts](#).
7. **Program the acceptance filters.** Refer to section [18.2.5 Rx Message Filtering](#).
8. **Select operating mode.** Normal, Sleep, Snoop or LoopBack. Refer to section [18.3.4 Change Operating Mode](#).
9. **Enable the controller.** Write a 1 to can.SRR[CEN].

18.3.4 Change Operating Mode

Example: Normal to Sleep Mode

Sleep mode is entered from Normal Mode when the following conditions are met:

1. **Select Sleep Mode.** Write 1 to can.MSR[SLEEP].
2. **Wait for CAN bus to go idle.**
3. **Wait for all TxFIFO and TxHPB messages to be transmitted.**

Note: In normal mode, can.MSR[LBACK] = 0 and can.SSR[CEN] = 1. Also, can.MSR[SNOOP] = don't care.

Example: Configuration to Sleep Mode

Sleep mode is entered from Configuration Mode when the following conditions are met:

1. **Select Sleep Mode.** Write 1 to can.MSR[SLEEP] and write 0 to can.MSR[LBACK].
2. **Enable the controller.** Write 1 to can.SSR[CEN].
3. **Wait for TxFIFO or TxHPB to empty.**

Note: In configuration mode, can.MSR[SNOOP] = don't care.

Sleep mode is exited when I/O bus activity is detected or when software writes a message to either the TxFIFO or the TxHPB. When the controller exits sleep mode, can.MSR[SLEEP] is set to 0 by the hardware and an interrupt can be generated.

18.3.5 Write Messages to TxFIFO

With either option, can.SR[TXFLL] can be polled before writing a message.

All messages written to the TxFIFO should follow the format defined in Message Structure.

Example: Write Message to TxFIFO Using Polling Method

1. **Poll the TxFIFO status.** Read can.SR[TXFLL] for 0 and can.SR[TXFEMP] for 1 and then message can be written into the TxFIFO.

2. **Write message to TxFIFO.** Write to all four data registers (can.TXFIFO_ID, can.TXFIFO_DLC, can.TXFIFO_DATA1, and can.TXFIFO_DATA2).

Example: Write Message to TxFIFO Using Interrupt Method

In interrupt mode, writes can continue until can.ISR[TXFL] generates an interrupt.

Messages can be continuously written to the TxFIFO until the TxFIFO is full. When the TxFIFO is full the can.ISR[TXFL] and can.SR[TXFL] are set to 1. When the TxFIFO is empty, can.ISR[TXFEMP] is set to 1.

18.3.6 Write Messages to TxHPB

All messages written to the TxHPB use the polling method. The format should follow section [18.2.2 Message Format](#).

Example: Write Message to TxHPB

1. **Poll the TxHPB status.** Read can.SR[TXBFL] until it equals 0 and then write the message into the TxHPB.
2. **Write message to TxHPB.** Write to all four data registers (can.TXFIFO_ID, can.TXHPB_DLC, can.TXHPB_DATA1, and can.TXHPB_DATA2).

18.3.7 Read Messages from RxFIFO

Whenever a new message is received and put into the RxFIFO, the can.ISR[RXNEMP] and can.ISR[RXOK] bits are set to 1. If the RxFIFO is empty when the message is read, then the can.ISR[RXUFLW] is also set to 1.

Example: Read Message from RxFIFO Using Polling Method

1. **Poll the RxFIFO status.** Read can.ISR[RXOK] or can.ISR[RXNEMP] register until a message is received. Proceed to step 2 when a bit is set.
2. **Read message from the RxFIFO.** Read all four of the registers (can.RXFIFO_ID, can.RXFIFO_DLC, can.RXFIFO_DATA1, can.RXFIFO_DATA2).
3. **Determine if more messages are in the RxFIFO.** Read can.ISR[RXNEMP].

Example: Read Message from RxFIFO Using Interrupt Method

The can.ISR[RXOK] and/or can.ISR[RXNEMP] bit fields can generate the interrupt.

1. **Program RxFIFO watermark level interrupt.** Write to can.WIR[FW] to set watermark can.ISR[RXFWMFL] interrupt.
2. **Proceed to step 3 when an interrupt is received.**
3. **Confirm content is waiting.** Read can.ISR[RXOK] or can.ISR[RXFWMFL].
4. **Read message from the RxFIFO.** Read all four of the registers (can.RXFIFO_ID, can.RXFIFO_DLC, can.RXFIFO_DATA1, can.RXFIFO_DATA2).

5. **Determine if more messages are in the RxFIFO.** Read can.ISR[RXNEMP].
6. **Repeat until the RxFIFO is empty.**
7. **Clear the interrupt.**

18.3.8 Register Overview

The control and status registers are listed in see [Table 18-6](#). Each of these registers is 32-bits wide. Any read operations to reserved bits or bits that are not used return 0. A 0 should be written to reserved bits and bit fields not used. Writes to reserved locations are ignored.

Table 18-6: CAN Register Overview

Function	Register Names (CAN registers, except where noted)	Overview
Configuration and Control	SRR MSR BRPR BTR ECR TCR	Enable/disable and reset the controller. Setup baud rate and timing. Clear timestamp counter.
Interrupt Processing	ISR IER ICR WIR	Enable/disable the interrupt detection, mark interrupt sent to the interrupt controller, read raw interrupt status.
Status	ECR ESR SR	Inform about the status of the controller.
Transmit FIFO	TXFIFO_ID TXFIFO_DLC TXFIFO_DATA1 TXFIFO_DATA2	Write message to be transmitted.
Transmit High Priority Buffer	TXHPB_ID TXHPB_DLC TXHPB_DATA1 TXHPB_DATA2	Store one high priority transmit message.
Receive FIFO	RXFIFO_ID RXFIFO_DLC RXFIFO_DATA1 RXFIFO_DATA2	Read received message.
Acceptance Filter	AFR AFMR[4:1] AFIR[4:1]	Configure and control the four acceptance filters.
System level	slcr.CAN_CLK_CTRL slcr.CAN_MIOCLK_CTRL slcr.CAN_RST_CTRL	A controller reset and clock control.

18.4 System Functions

18.4.1 Clocks

The controller and I/O interface are driven by the reference clock (CANx_REF_CLK). The controller's interconnect also requires an APB interface clock. The APB interconnect clock (CPU_1x) always comes from the PS clock subsystem.

The reference clock normally comes from the PS clock subsystem, but it can alternatively be driven by an external clock source via any available MIO pin. The reference clock is used by the protocol engine, the baud rate generator, and the datapath. The controllers share the same reference clock frequency from the PS clock subsystem. If the reference clock is from an MIO pin, then the frequencies can be different.

CPU_1x Clock

Refer to [Chapter 25, Clocks](#), for general clock programming information. The CPU_1x clock runs asynchronous to the CAN reference clock.

Reference Clock

CAN_REF_CLK is normally sourced from the PS clock subsystem, but it can alternatively be driven by an external clock source via an MIO pin. Internally, the PS has three PLLs and two clock divider pairs. The clock source choice, PS clock subsystem or external MIO pin, is controlled by the CAN_MIOCLK_CTRL register.

The CAN clocks in the PS are controlled by slcr.CAN_CLK_CTRL. The generation of the CAN reference clock by the PS is described in section [25.7.4 CAN Clocks](#). There is one clock generator in the PS for both CAN controllers. If an MIO pins is used instead, the selected MIO_PIN Mux register is programmed as an input.

Example: Configure and Route Internal Clock for Reference Clock

Configure the clock and disable MIO path. Assume the PLL is operating at 1000 MHz and the required CAN reference clock is 24 MHz (23.8095 MHz).

1. **Program the clock subsystem.** Write 0x0030_0E03 to the slcr.CAN_CLK_CTRL register:
 - a. Enable both CAN reference clocks.
 - b. Divide the I/O PLL clock by 42 (0x02A): DIVISOR0= 0x0E and DIVISOR1=0x03 used by both controllers.
2. **Disable the MIO path.** Write 0x0000_0000 to the slcr.CAN_MIOCLK_CTRL register to select the clock from the internal clock subsystem/PLL for both controllers.

Example: Source Controller Clock from MIO Pin

This example uses MIO pin 45 as a controller clock reference.

1. **Configure MIO device pin.** Write 0x0000_1200 to the slcr.MIO_PIN_45 register:
 - a. Route MIO pin 45 to the GPIO controller (this is overridden in the next step).
 - b. Disable the output driver (TRI_ENABLE = 1).
 - c. LVCMOS18 (refer to the register definition for other voltage options).
 - d. Slow CMOS edge (benign setting).
 - e. Enable internal pull-up resistor.
 - f. Disable HSTL receiver.
2. **Enable MIO path.** Write to the slcr.CAN_MIOCLK_CTRL register to override the MIO PIN register setting that was written in the previous step.

Write a 1 to slcr.CAN_MIOCLK_CTRL[CANx_REF_SEL] and write the desired MIO pin number into the slcr.CAN_MIOCLK_CTRL[CANx_MUX] bit field to match the pin in the previous step.

18.4.2 Resets

The effects for each reset type are summarized in [Table 18-7](#).

Table 18-7: CAN Reset Effects

Name	APB Interface	Rx and Tx FIFOs	Protocol Engine	Control and Status Registers	Acceptance Filters (ID and Mask)
Local CAN Reset can.SRR[SRST]	Yes	Yes	Yes	Yes	No
PS Reset Subsystem slcr.CAN_RST_CTRL[CANx_CPU1X_RST]	Yes	Yes	Yes	Yes	No

Example: Reset using Local CAN Reset

Write to the Local CAN reset register. Write a 1 to can.SRR[SRST] bit field. This bit is self-clearing.

Example: Reset using Reset Subsystem

Write to the slcr reset register for CAN. Write a 1 then a 0 to the slcr.CAN_RST_CTRL[CANx_CPU1X_RST] bit field.

Each set of controller Rx/Tx signals is connected to either MIO pins or the EMIO interface, refer to [Table 18-8, page 448](#). The general routing concepts and MIO I/O buffer configurations are explained in section [2.4 MIO-EMIO](#).

1. **Configure MIO pin 46 for the Rx signal.** Write 0x0000_1221 to the slcr.MIO_PIN_46 register:
 - a. Route CAN0 Rx signal to pin 46.
 - b. Output disabled (set TRI_ENABLE = 1).
 - c. LVCMOS18 (refer to the register definition for other voltage options).
 - d. Slow CMOS edge (benign setting).

- e. Enable internal pull-up resistor.
 - f. Disable HSTL receiver.
2. **Configure MIO pin 47 for the Tx signal.** Write `0x0000_1220` to the `slcr.MIO_PIN_47` register:
 - a. Route CAN0 Tx signal to pin 47.
 - b. 3-state controlled by CAN (`TRI_ENABLE = 0`).
 - c. LVCMOS18 (refer to the register definition for other voltage options).
 - d. Slow CMOS drive edge.
 - e. Enable internal pull-up resistor.
 - f. Disable HSTL receiver.

18.5 I/O Interface

18.5.1 MIO Programming

Each set of controller Rx/Tx signals is connected to either MIO pins or the EMIO interface, refer to [Table 18-8, page 448](#). The general routing concepts and MIO I/O buffer configurations are explained in section [2.4 MIO-EMIO](#). The MIO routing to use an external reference clock (`CAN_REF_CLK`) is described in section [18.4.1 Clocks](#).

Example: Configure Rx/Tx Signals to MIO Pins

1. **Configure MIO pin 46 for the Rx signal.** Write `0x0000_1221` to the `slcr.MIO_PIN_46` register:
 - a. Route CAN0 Rx signal to pin 46.
 - b. Output disabled (set `TRI_ENABLE = 1`).
 - c. LVCMOS18 (refer to the register definition for other voltage options).
 - d. Slow CMOS edge (benign setting).
 - e. Enable internal pull-up resistor.
 - f. Disable HSTL receiver.
2. **Configure MIO pin 47 for the Tx signal.** Write `0x0000_1220` to the `slcr.MIO_PIN_47` register:
 - a. Route CAN0 Tx signal to pin 47.
 - b. 3-state controlled by CAN (`TRI_ENABLE = 0`).
 - c. c. LVCMOS18 (refer to the register definition for other voltage options).
 - d. d. Slow CMOS drive edge.
 - e. e. Enable internal pull-up resistor.
 - f. f. Disable HSTL receiver.

18.5.2 MIO-EMIO Signals

The CAN I/O Signals are identified in [Table 18-8](#). Refer to section [2.4 MIO-EMIO](#) for routing details. The MIO pins and any restrictions based on device versions are shown in the MIO table in section [2.4.4 MIO-at-a-Glance Table](#).

Table 18-8: CAN MIO Pins and EMIO Signals

CAN Interface	Default Controller Input Value	MIO Pins		EMIO Signals	
		Numbers	I/O	Name	I/O
CAN 0 Rx	0	10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50	I	EMIOCAN0PHYRX	I
CAN 0 Tx	~	11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51	O	EMIOCAN0PHYTX	O
CAN 0 CLK	~	Any MIO pin	I	~	~
CAN 1 Rx	0	9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53	I	EMIOCAN1PHYRX	I
CAN 1 Tx	~	8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52	O	EMIOCAN1PHYTX	O
CAN 1 CLK	~	Any MIO pin	I	~	~

UART Controller

19.1 Introduction

The UART controller is a full-duplex asynchronous receiver and transmitter that supports a wide range of software programmable baud rates and data formats. It can also accommodate automatic parity generation and several error detection schemes, and provides both data buffering in both the receive FIFO (RxFIFO) and the transmit FIFO (TxFIFO). The PS provides two UART controllers, UART_x where $x = 0$ or 1 .

The UART controller is structured into separate receiver and transmitter data paths, which include 64-byte FIFOs. The rate of operation of these data paths is controlled by the baud rate generator module.

The mode of operation is configured using the control logic module, and the current status of the UART is indicated via the interrupt control module. The current mode is also used to control the mode switch module which selects the various loopback modes available.

Data to be transmitted is written into the TxFIFO using byte operations via the APB interface.

When the TxFIFO contains enough data to transmit, the transmitter module pulls the data from the FIFO and serializes it onto the transmitter serial output.

Data received is deserialized by the receiver module and written into the RxFIFO. The fill level of the RxFIFO is then used to trigger an interrupt. Software can read byte and double-byte data from the RxFIFO..

If the UART is being used in a modem-like application, the modem control module detects and generates the modem handshake signals appropriately and also controls the receiver and transmitter paths according to the handshaking protocol.

19.1.1 Features

The PS supports two UART devices in the IOP with these key features:

- Programmable baud rate generator
- 64-byte receive and transmit FIFOs
- Programmable protocol:
 - 6, 7, or 8 data bits

- 1, 1.5, or 2 stop bits
- Odd, even, space, mark, or no parity
- Parity, framing and overrun error detection
- Line-break generation and detection
- Interrupts generation
- Rx/D and Tx/D modes: Normal/echo and diagnostic loopbacks Rx/D and Tx/D can be routed to MIO pins or the EMIO interface.
- Modem control signals: CTS, RTS, DSR, DTR, RI and DCD are available on the EMIO interface

19.1.2 System Viewpoint

The system viewpoint diagram for the UART module is shown in Figure 19-1.

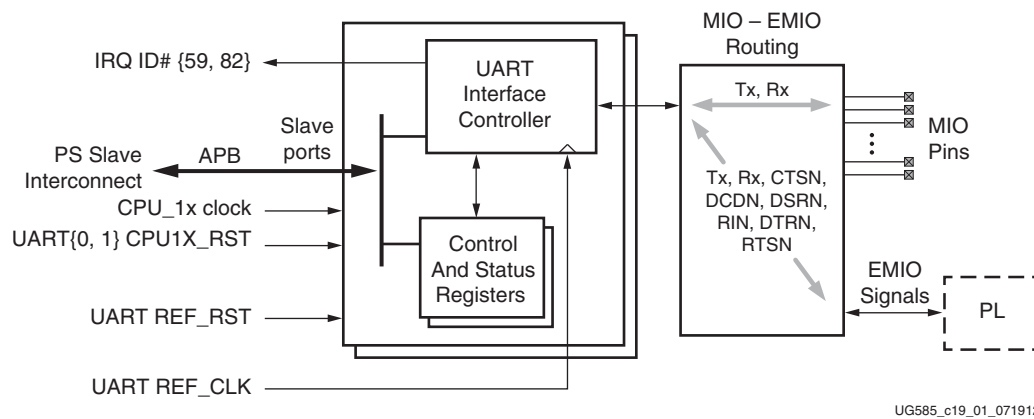


Figure 19-1: UART System Viewpoint

There are two UARTs in the PS. Each UART can be configured and operated independently. The TX and RX interface signals can be mapped to either the EMIO interface or the MIO interface. The modem control signals are only available through the EMIO interface. The clocks and resets to the UART are sourced from the global clock and reset blocks for the PS. The controller also has local resets for the RxFIFO and TxFIFO. The APB interface allows multiple masters to have access to the controllers. The software-driven modem flow control signals can alternatively be implemented via the GPIO controller and can be routed to either the MIO or EMIO.

19.1.3 Block Diagram

The block diagram for the UART module is shown in [Figure 19-2](#)

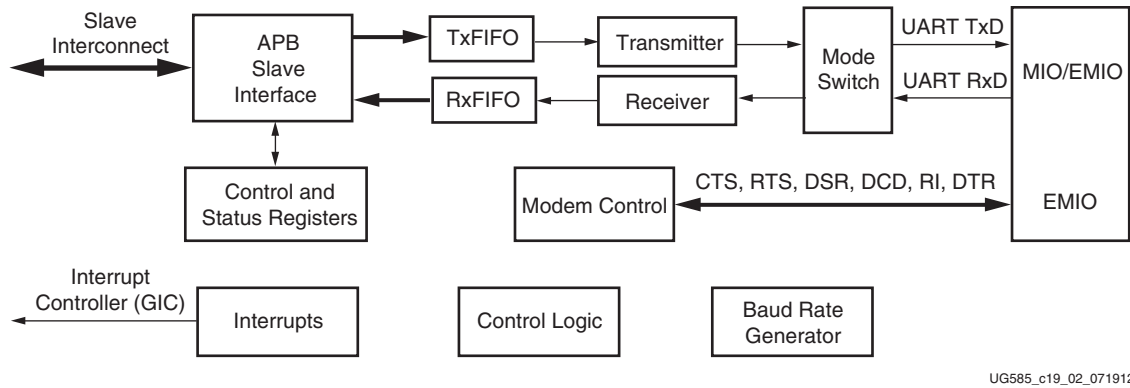


Figure 19-2: UART Block Diagram

Control Logic

The control logic contains the Control register and the Mode register, which are used to select the various operating modes of the UART.

The Control register enables, disables, and issues soft resets to the receiver and transmitter modules. In addition, it restarts the receiver timeout period, and controls the transmitter break logic.

Receive line break detection must be implemented in Software. It will be indicated by a Frame Error followed by one or more zero bytes in the Rx FIFO.

The Mode register selects the clock used by the baud rate generator. It also selects the bit length, parity bit and stop bit to be used by transmitted and received data. In addition, it selects the mode of operation of the UART, switching between normal UART mode, automatic echo, local loopback, or remote loopback, as required.

19.1.4 Notices

Operating Restrictions

There is a single PS clock generator for both UART controllers. The reference clocks going to the baud rate generator are of the same clock frequency, see [section 25.7.3 SDIO, SMC, SPI, Quad-SPI and UART Clocks](#). The controllers are always clocked by the internal, PS clock generator.

19.2 Functional Description

19.2.1 Baud Rate Generator

The baud rate generator furnishes the bit period clock, or baud rate clock, for both the receiver and the transmitter. The baud rate clock is implemented by distributing the base clock `uart_clk` and a single cycle clock enable to achieve the effect of clocking at the appropriate frequency division. The effective logic for the baud rate generation is shown in Figure 19-3.

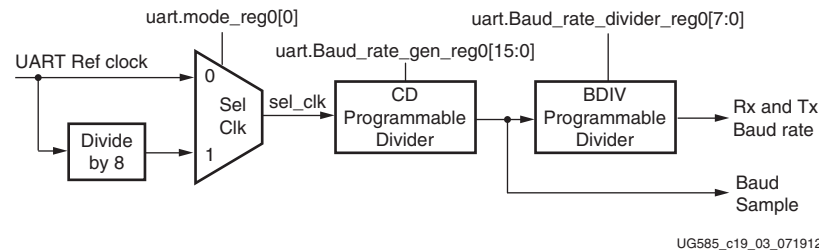


Figure 19-3: UART Board Rate Generator

The baud rate generator can use either the master clock signal, `uart_ref_clk`, or the master clock divided by eight, `uart_ref_clk/8`. The clock signal used is selected according to the value of the CLKS bit in the Mode register (`ua_mr`). The resulting selected clock is termed `sel_clk` in the following description.

The `sel_clk` clock is divided down to generate three other clocks: `baud_sample`, `baud_tx_rate`, and `baud_rx_rate`. The `baud_tx_rate` is the target baud rate used for transmitting data. The `baud_rx_rate` is nominally at the same rate, but gets resynchronised to the incoming received data. The `baud_sample` runs at a multiple of `baud_rx_rate` and `baud_tx_rate` and is used to over-sample the received data.

The `sel_clk` clock frequency is divided by the CD field value in the Baud Rate Generator register to generate the `baud_sample` clock enable. This register may be programmed with a value between 1 and 65535.

The `baud_sample` clock is divided by BDIV plus 1. BDIV is a programmable field in the Baud Rate Divider register and can be programmed with a value between 0 and 255. It has a reset value of 15, inferring a default ratio of 16 `baud_sample` clocks per `baud_tx_clock` / `baud_rx_rate`.

Thus the frequency of the `baud_sample` clock enable is shown in Equation 19-1.

$$\text{baud_sample} = \frac{\text{sel_clk}}{CD} \quad \text{Equation 19-1}$$

The frequency of the baud_rx_rate and baud_tx_rate clock enables is show in Equation 19-2.

$$baud_rate = \frac{sel_clk}{CD \times (BDIV + 1)} \quad \text{Equation 19-2}$$

It is essential to disable the transmitter and receiver before writing to the Baud Rate Generator register (ua_brgr), or the baud rate divider register (ua_div). A soft reset must be issued to both the transmitter and receiver before they are re-enabled.

Some examples of the relationship between the uart_ref_clk clock, baud rate, clock divisor, and the rate of error are shown in Table 19-1. The highlighted entry shows the default reset values for CD and BDIV. For this example a system clock rate of Uart_ref_clk = 50 MHz and Uart_ref_clk/8 = 6.25 MHz is assumed. See Chapter 25, Clocks for details on system clock generation.

Table 19-1: UART Parameter Value Examples

Clock	Baud Rate	Calculated CD	Actual CD	BDIV	Actual Baud Rate	Error (BPS)	% Error
UART Ref clock	600	10416.667	10417	7	599.980	0.020	-0.00003
UART Ref clock /8	9,600	81.380	81	7	9645.061	45.061	0.46
UART Ref clock	9,600	651.041	651	7	9600.614	0.614	0.00006
UART Ref clock	28,800	347.222	347	4	28818.44	18.44	0.00064
UART Ref clock	115,200	62	62	6	115207.3733	7.3733	0.0064
UART Ref clock /8	230,400	5.425	5	4	250000	19600	0.08506

19.2.2 Transmit FIFO

The transmit FIFO (TxFIFO) stores data written from the APB interface until it is removed by the transmit module and loaded into its shift register. The TxFIFO's maximum data width is eight bits. Data is loaded into the TxFIFO by writing to the TxFIFO register.

When data is loaded into the TxFIFO, the TxFIFO empty flag is cleared and remains in this Low state until the last word in the TxFIFO has been removed and loaded into the transmitter shift register. This means that host software has another full serial word time until the next data is needed, allowing it to react to the empty flag being set and write another word in the TxFIFO without loss in transmission time.

The TxFIFO full flag indicates that the TxFIFO is completely full and prevents any further data from being loaded into the TxFIFO. If another APB write to the TxFIFO is performed, an overflow is triggered and the write data is not loaded into the TxFIFO.

The TxFIFO nearly-full flag indicates that there is only byte free in the TxFIFO.

A threshold trigger can be setup on the TxFIFO fill level. The Transmitter Trigger register can be used to setup this value, such that the trigger is set when the TxFIFO fill level reaches this programmed value.

19.2.3 Transmitter Data Stream

The transmit module removes parallel data from the TxFIFO and loads it into the transmitter shift register so that it can be serialized.

The transmit module shifts out the start bit, data bits, parity bit, and stop bits as a serial data stream. Data is transmitted, least significant bit first, on the falling edge of the transmit baud clock enable (baud_tx_rate). A typical transmitted data stream is illustrated in Figure 19-4.

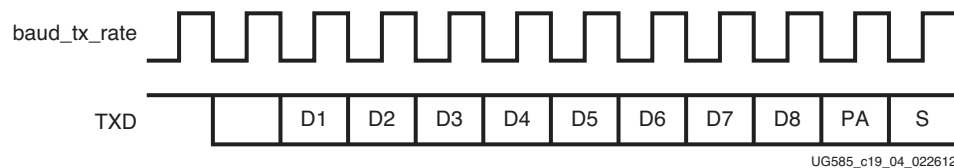


Figure 19-4: Transmitted Data Stream

The CHRL in the Mode register selects the character length, in terms of the number of data bits. The NBSTOP field in the Mode register selects the number of stop bits to transmit.

19.2.4 Receiver FIFO

The RxFIFO stores data that is received by the receiver serial shift register. The RxFIFO's maximum data width is eight bits.

When data is loaded into the RxFIFO, the RxFIFO empty flag is cleared and this state remains Low until all data in the RxFIFO has been transferred through the APB interface. Reading from an empty RxFIFO returns zero.

The RxFIFO full flag (rfifo_full) indicates that the RxFIFO is full and prevents any further data from being loaded into the RxFIFO. When a space becomes available in the RxFIFO, any character stored in the receiver will be loaded.

A threshold trigger can be setup on the RxFIFO fill level. The Receiver Trigger register can be used to setup this value, such that the trigger is set when the RxFIFO fill level reaches this programmed value.

19.2.5 Receiver Data Capture

The UART continuously over-samples the UARTx_RxD signal using UARTx REF_CLK and the clock enable (baud_sample). When the samples detect a transition to a Low level, it can indicate the beginning of a start bit. When the UART senses a Low level at the UART_RxD input, it waits for a count of half of BDIV baud rate clock cycles, and then samples three more times. If all three bits still indicate a Low level, the receiver considers this to be a valid start bit, as illustrated in Figure 19-5 for the default BDIV of 15.

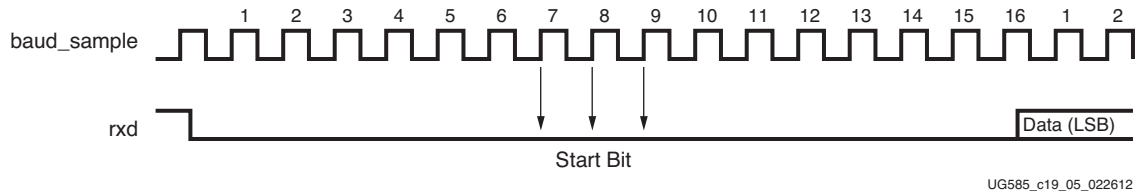


Figure 19-5: Default BDIV Receiver Data Stream

When a valid start bit is identified, the receiver baud rate clock enable (baud_rx_rate) is re-synchronised so that further sampling of the incoming ua_rxd occurs around the theoretical mid-point of each bit, as illustrated in Figure 19-6.

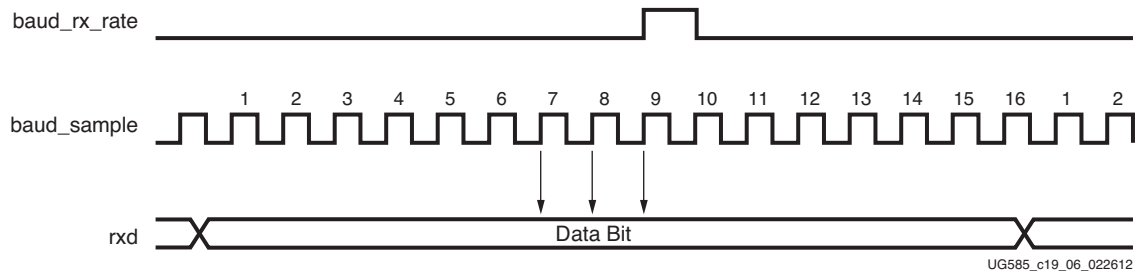


Figure 19-6: Re-synchronized Receiver Data Stream

When the re-synchronised baud_rx_rate is High, the last three sampled bits are compared. The logic value is determined by majority voting; two samples having the same value define the value of the data bit. When the value of a serial data bit has been determined, it is shifted to the receive shift register. When a complete character has been assembled, the contents of the register are then pushed to the Rx FIFO.

Receiver Parity Error

Each time a character is received, the receiver calculates the parity of the received data bits in accordance with the UART.mode_reg0[PAR] bit field. It then compares the result with the received parity bit. If a difference is detected, the parity error bit is set = 1, UART.Channel_sts_reg0[PARE]. An interrupt is generated, if enabled.

Receiver Framing Error

When the receiver fails to receive a valid stop bit at the end of a frame, the frame error bit is set = 1, UART.Channel_sts_reg0[FRAME]. An interrupt is generated, if enabled.

Receiver Overflow Error

When a character is received, the controller checks to see if the Rx FIFO has room. If it does, then the character is written into the Rx FIFO. If the Rx FIFO is full, then the controller waits. If a subsequent start bit on RxD is detected and the Rx FIFO is still full, then data is lost and the controller sets the Rx overflow interrupt bit, UART.Channel_sts_reg0[ROVR] = 1. An interrupt is generated, if enabled.

19.2.6 Mode Switch

The mode switch controls the routing of the Rx/D and Tx/D signals between the controller and the MIO/EMIO. There are four operating modes as shown in Figure 19-7. The mode is controlled by the `uart.mode_reg0[CHMODE]` register bit field: normal, automatic echo, local loopback and remote loopback.

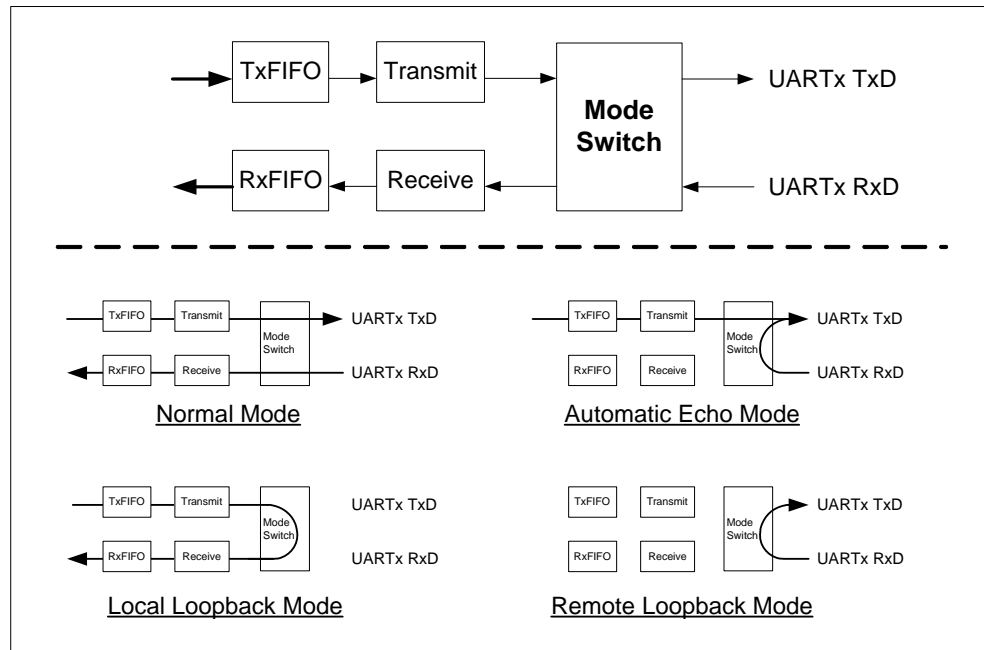


Figure 19-7: UART Mode Switch for Tx/D and Rx/D

Normal Mode

Normal mode is used for standard UART operation. This is shown in Figure 19-7.

Automatic Echo Mode

Echo mode receives data on Rx/D and the mode switch routes the data to both the receiver and the Tx/D pin. Data from the transmitter cannot be sent out from the controller.

Local Loopback Mode

Local loopback mode does not connect to the Rx/D or Tx/D pins. Instead, the transmitted data is looped back around to the receiver.

Remote Loopback Mode

Remote loopback mode connects the Rx/D signal to the Tx/D signal. In this mode, the controller cannot send anything on Tx/D and the controller does not receive anything on Rx/D.

19.2.7 Modem Control

The modem control module facilitates the control of communication between a modem and the UART. It contains the Modem Status register and the Modem Control register.

The read-only Modem Status register is used to read the values of the clear to send, data carrier detect, data set ready, and ring indicator modem inputs. It also reports changes in any of these inputs and indicates whether automatic flow control mode is currently enabled. The bits in the Modem Status register are cleared by writing a High to the particular bit.

The read/write only Modem Control register is used to set the data terminal ready and request to send outputs, and to enable the Automatic Flow Control Mode register.

By default, the automatic flow control mode is disabled, meaning that the modem inputs and outputs work completely under software control. When the automatic flow control mode is enabled by setting the FCM bit in the Modem Control register, the UART transmission and reception status is automatically controlled using the modem handshake inputs and outputs.

In automatic flow control mode the request to send output is asserted and de-asserted based on the current fill level of the receiver FIFO, which results in the far-end transmitter pausing transmission and preventing an overflow of the UART receiver FIFO. The FDEL field in the Flow Delay register (Flow_delay_reg0) is used to setup a trigger level on the Receiver FIFO which causes the de-assertion of the request to send. It remains Low until the FIFO level has dropped to below four less than FDEL.

Additionally in automatic flow control mode, the UART only transmits while the clear to send input is asserted. When the clear to send is de-asserted, the UART pauses transmission at the next character boundary.

Example: Automatic Flow Control

1. **Select automatic flow control.** Write a 1 to `uart.Modem_ctrl_reg0[FCM]`.
2. **Set RTS trigger level.** Write to the `uart.Flow_Delay_reg0` register. This is the trigger level to de-assert modem signal RTS.
3. **Verify the mode change to automatic.** Read `uart.Modem_sts_reg0[FCMS]` until it equals 1.

When software writes a 1 to `uart.Modem_ctrl_reg0[FCM]`, the modem changes to automatic mode. The change of mode from manual to automatic is verified by reading the status bit FCMS in the Modem Status register.

19.3 Programming Guide

Example: Programming Steps

1. Software Reset (see [19.4.2 Resets](#))
2. Configure I/O Signal Routing and MIO Pins (see section [19.5.1 MIO Programming](#))
3. Configure Clock (see section [19.4.1 Clocks](#))

4. Configure Baud Rate Generator (see section [19.2.1 Baud Rate Generator](#))
5. Modem Control (see section [19.3.2 Modem Flow Control](#))
6. Interrupt Control (see section [19.3.4 Interrupt Control](#))
7. Rx and Tx Data Transfers (see section [19.3.3 Rx and Tx Data Transfers](#))

19.3.1 Configuration Register

Example: Enable Tx/RX and Configure I/O Protocol (Control_reg0 register)

Write to the UART control register (uart.Control_reg0) to control enable/disable and reset the transmit and receive data paths. Also, use this register to control on transmission of start/stop of break character and on restart of receiver timeout counter.

1. **Enable the Transmitter.** Write 1 to uart.Control_reg0[TXEN] and 0 to uart.Control_reg0[TXDIS].
2. **Enable the Receiver.** Write 1 to uart.Control_reg0[RXEN] and 0 to uart.Control_reg0[RXDIS].
3. **Reset the Transmitter and Receiver data paths.** Write a 1 to uart.Control_reg0[TXRES] and uart.Control_reg0[RXRES] and wait until the reset is completed as these bits are self-clearing.
4. **Disable the break character transmission.** Write a 1 to uart.Control_reg0[STPBRK] to disable break character transmission. Otherwise, clear this bit and set uart.Control_reg0[STTBK] to 1.
5. **Reset receiver timeout counter.** Write a 1 to uart.Control_reg0[RSTTO] to clear the receiver timeout counter.

19.3.2 Modem Flow Control

The UART flow control signals, DTR and RTS are generated by the UART controller.

- The RTS flow control signal is used to signal for Rx (ready to send signal to the attached terminal).
- The DTR flow control signal indicates the status of the UART controller (data terminal ready).

The DTR and RTS can be controlled manually by software or automatically by the controller.

- In automatic mode, the modem control unit asserts and de-asserts the RTS and DTR signals.
- In manual mode, software controls the RTS and DTR signals using uart.Modem_ctrl_reg0.

uart.Modem_ctrl_reg0:

- [DTR]: Data Terminal Ready output signal
- [RTS]: Request to Send output signal
- [FCM]: Select Automatic or Manual flow control.

uart.Modem_sts_reg0:

- [DCTS]: Delta Clear To Send (input) status
- [DDSR]: Delta Data Set Ready (input) status

- [TERI]: Trailing-edge Ring Indicator (input) status

Select one of the following operating options:

Option: Manual Flow Control

1. **Select manual flow control.** Write a 0 to `uart.Modem_ctrl_reg0[FCM]`.
 - a. Control the DTR output signal using `uart.Modem_ctrl_reg0[DTR]`.
 - b. Control the RTS output signal using `uart.Modem_ctrl_reg0[RTS]`.

Example: Monitor for a Change in the DCD DSR RI CTS Flow Control Signals

A logic level change to the DCD DSR RI CTS flow control signals is detected by the controller. When a logic level change is detected, the hardware sets the `uart.Channel_sts_reg0[DMSI]` bit. This change, or channel status can optionally generate an interrupt.

1. **Check flow control signal status.** `uart.Modem_sts_reg0` register reports the channel status.
 - a. In polled mode, software can poll `uart.Channel_sts_reg0[DMSI]` to detect any changes of status in modem signals.
 - b. In interrupt mode, the ISR can run when the DMSI interrupt occurs when there is a change of status on modem lines.

19.3.3 Rx and Tx Data Transfers

Software can use polling or interrupts to control the flow of data to the TxFIFO and RxFIFO.

Example: Transmit Data using the Polling Method

In this example, the software can choose to fill the TxFIFO until the Full status bit is set or wait for the TxFIFO to be empty (and write up to 64 bytes).

1. **Check to see if the TxFIFO is empty.** Read until `uart.Channel_sts_reg0[EMPTY] = 1`.
2. **Fill the TxFIFO.** Write data to the `uart.TX_RX_FIFO0` register.
3. **Check for space in FIFO or wait for FIFO to be empty.**

Option A: Check to make sure the TxFIFO is not full before adding more data. Read `uart.Channel_sts_reg0[TFUL]` until it equals 1. Write a single byte of data to the TxFIFO and read TFUL again.

Option B: Wait until the TxFIFO goes empty. Read `uart.Channel_sts_reg0[EMPTY]` until it equals 1, then go to step 2 to fill the TxFIFO.

Example: Transmit Data using the Interrupt Method

1. **Disable the interrupt.** Write a 1 to `uart.Intrpt_dis_reg0[EMPTY]`.
2. **Fill the TxFIFO.** Write data to the `uart.TX_RX_FIFO0` register.
3. **Check for space in FIFO.** Check that `uart.Channel_sts_reg0[TFUL] = 1` before adding more data.

4. **Repeat step 2 and 3.** Repeat until `uart.Channel_sts_reg0[TFUL]` is not set.
5. **Enable the interrupt.** Enable the interrupt with a write of 1 to `uart.Intrpt_en_reg0[EMPTY]`.
6. **Wait until the TxFIFO is empty.** Repeat from step 1 when `uart.Channel_int_sts_reg0[EMPTY]` is set to 1.

Example: Tx Trigger Interrupt Method

1. **Disable the interrupt.** Write a 1 to `uart.Intrpt_dis_reg0[TTRIG]` bit.
2. **Fill the TxFIFO.** Write data to the `uart.TX_RX_FIFO0` register.
3. **Check for space in FIFO.** Check that `uart.Channel_sts_reg0[TFUL] = 1` before adding more data.
4. **Repeat step 2 and 3.** Repeat until `uart.Channel_sts_reg0[TFUL]` is not set.
5. **Enable the interrupt.** Enable the interrupt `uart.Intrpt_en_reg0[TTRIG]`.
6. **Wait until the TxFIFO is empty.** Repeat from step 1 when `uart.Channel_int_sts_reg0[TTRIG]` is set to 1.

Example: Receive Data using the Polling Method

1. **Wait until the RxFIFO is filled up to the trigger level.** Check to see if `uart.Channel_sts_reg0[RTRIG] = 1`.
2. **Read data from the RxFIFO.** Read data from the `uart.TX_RX_FIFO0` register.
3. **Check for errors on the received data.** Check for a 1 on the error status bits `uart.Channel_sts_reg0[FRAME]`, `uart.Channel_sts_reg0[PARE]`, `uart.Channel_sts_reg0[ROVR]` and `uart.Channel_sts_reg0[TIMEOUT]`.
4. **Repeat step 2 and 3 until the FIFO is empty.** Check that `uart.Channel_sts_reg0[EMPTY] = 1`.

Example: Receive Data using the Interrupt Method

1. **Enable interrupts.** Write a 1 to `uart.Intrpt_en_reg0[EMPTY]` and `uart.Intrpt_en_reg0[RTRIG]`.
2. **Wait until the RxFIFO is filled up to the trigger level.** Check that `uart.Channel_int_sts_reg0[RTRIG] = 1`.
3. **Read data from the RxFIFO.** Read data from `uart.TX_RX_FIFO0` register.
4. **Check for errors on the received data.** Check for a 1 on error status bits `uart.Channel_int_sts_reg0[FRAME]`, `uart.Channel_int_sts_reg0[PARE]`, `uart.Channel_int_sts_reg0[ROVR]` and `uart.Channel_int_sts_reg0[TIMEOUT]`.
5. **Repeat step 2 and 3 until the FIFO is empty.** Check that `uart.Channel_sts_reg0[EMPTY] = 1`.

19.3.4 Interrupt Control

The interrupt registers and bit fields are summarized in [Table 19-2](#).

Table 19-2: UART Interrupt Status Bits

Interrupt Register Names and Bit Assignments														
14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
uart.Intrpt_en_reg0														
uart.Intrpt_dis_reg0														
uart.Intrpt_mask_reg0														
uart.Chnl_int_sts_reg0														
x	x	TOVR	TNFUL	TTRIG	DMSI	TIME OUT	PARE	FRAME	ROVR	TFUL	EMPTY	RFUL	EMPTY	RTRIB
uart.Channel_sts_reg0														
TNFUL	TTRIG	FDELT	TACTIVE	RACTIVE	DMSI	TIME OUT	PARE	FRAME	ROVR	TFUL	EMPTY	RFUL	EMPTY	RTRIB

Interrupt Status

The two status registers can be read by software and reveal the raw and masked version of the interrupts. The masked versions are OR'd together assert the interrupt to the GIC interrupt controller. Interrupts are generated by the hardware and each bit is sticky. Software clears an interrupt by writing a 1 to the associated bit in the raw interrupt status register, Chnl_int_sts_reg0.

- **Chnl_int_sts_reg0**: Read: raw interrupt status bits. Write: clear raw interrupt bit (W1C).
- **Channel_sts_reg0**: Read-only, interrupt status after the enable/mask. These bits are OR'd together to assert the IRQ signal to GIC.

Interrupt Mask /Enable

Intrpt_mask_reg0 is a read-only interrupt mask/enable register that is used to mask individual raw interrupts:

- If the interrupt bit = 0, then the interrupt is masked.
- If the interrupt bit = 1, then the interrupt is enable.

This mask is controlled by the write-only Intrpt_en_reg0 and Intrpt_dis_reg0 registers. Each associated interrupt bit should be set mutually exclusive (one is set = 1 and the other is cleared = 0).

Channel Status

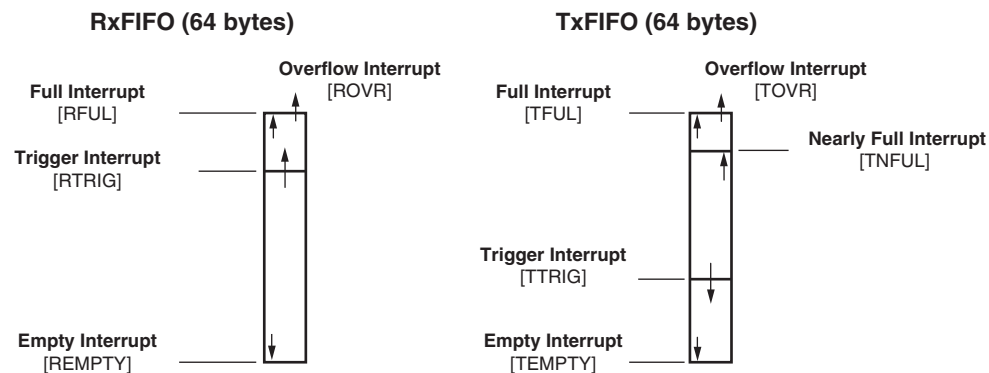
- **TACTIVE**: Transmitter state machine active status. If in an active state, the transmitter is currently shifting out a character.
- **RACTIVE**: Receiver state machine active status. If in an active state, the receiver is has detected a start bit and is currently shifting in a character.
- **FDELT**: Receiver flow delay trigger continuous status.

Non-FIFO Interrupts

- **TIMEOUT:** Receiver Timeout Error interrupt status. This event is triggered whenever the receiver timeout counter has expired due to a long idle condition.
- **PARE:** Receiver Parity Error interrupt status. This event is triggered whenever the received parity bit does not match the expected value.
- **FRAME:** Receiver Framing Error interrupt status. This event is triggered whenever the receiver fails to detect a valid stop bit.
- **DMSI:** indicates a change of logic level on the DCD, DSR, RI or CTS modem flow control signals.

FIFO Interrupts

The status bits for the FIFO interrupts listed in Table 19-2 are illustrated in Figure 19-8.



UG585_c19_12_072312

Figure 19-8: UART RxFIFO and TxFIFO Interrupt Positions

The FIFO trigger levels are controlled by these bit fields:

- `uart.Tx_FIFO_trigger_level0[TTRIG]`, a 6-bit field.
- `uart.Rcvr_FIFO_trigger_level0[RTRIG]`, a 6-bit field.

Example: Set the RxFIFO Trigger Level and Enable the Interrupt

The `Intrpt_en_reg0` register has bits to enable the interrupt mask and `Intrpt_dis_reg0` has bits to forcefully disable the interrupts. Each pair of bits should be set mutually exclusive (i.e., one register has a 1 for the bit and the other register has a 0):

- `Intrpt_en_reg0`: Write-to-set. Enable interrupt bit(s).
- `Intrpt_dis_reg0`: Write-only. Force disable of interrupt bit(s).

1. **Program the Trigger Level.** Write to the 6-bit field, `uart.Rcvr_FIFO_trigger_level0[RTRIG]`.
2. **Enable the RTRIG interrupt.** Set the enable bit and clear the disable bit:
 - a. Set `uart.Intrpt_en_reg0[RTRIG] = 1`.
 - b. Clear `uart.Intrpt_dis_reg0[RTRIG] = 0`.

- c. The `uart.intrpt_mask_reg0[RTRIG]` read back = 1.
3. **Disable the RTRIG interrupt.** Set the disable bit and clear the enable bit:
 - a. Set `uart.Intrpt_dis_reg0[RTRIG]` = 1.
 - b. Clear `uart.Intrpt_en_reg0[RTRIG]` = 0.
 - c. The `uart.intrpt_mask_reg0[RTRIG]` read back = 0.
4. **Clear the RTRIG interrupt.** Write a one to the `uart.Intrpt_dis_reg0[RTRIG]` bit field.

When both the enable and disable bits are set for an interrupt, the interrupt is disabled.

The state of the interrupt enable/disable mechanism can be determined by reading the `uart.Intrpt_mask_reg0` register. If the mask bit = 1, then the interrupt is enabled.

19.3.5 Register Overview

An overview of the UART registers is shown in [Table 19-3](#). Details are provided in section [B.33 UART Controller \(UART\)](#) of [Appendix B, Register Details](#).

Table 19-3: UART Register Overview

Function	Register Names	Overview
Configuration	Control_reg0 mode_reg0 Baud_rate_gen_reg0 Baud_rate_divider_reg0	Configure mode and baud rate. Read status.
Interrupt processing	Intrpt_en_reg0 Intrpt_dis_reg0 Intrpt_mask_reg0 Chnl_int_sts_reg0 Channel_sts_reg0	Enable/disable the interrupt detection, mask interrupt sent to the interrupt controller, read raw interrupt status for: TxFIFO Overflow, TxFIFO Nearly Full, TxFIFO Trigger, Delta Modem Status Indicator, Receiver Timeout Error, Receiver Parity Error, Receiver Framing Error, Receiver Overflow Error, Transmitter FIFO Full, Transmitter FIFO Empty, Receiver FIFO Full, Receiver FIFO Empty, and Receiver FIFO Trigger.
Receiver	Rcvr_timeout_reg0 Rcvr_FIFO_trigger_level0	Configure receiver timeout and FIFO trigger level value.
Transmitter	Tx_FIFO_trigger_level0	Configure TxFIFO trigger level value.
Modem	Modem_ctrl_reg0 Modem_sts_reg0 Flow_delay_reg0	Configure modem-like application.
Data	TX_RX_FIFO0	Read data Received. Write data to be Transmitted.

19.4 System Functions

19.4.1 Clocks

The controller and I/O interface are driven by the reference clock (UART_REF_CLK). The controller's interconnect also requires an APB interface clock (CPU_1x). Both of these clocks always come from the PS clock subsystem.

CPU_1x Clock

Refer to section [25.3 CPU Clock Domains](#), for general clock programming information. The CPU_1x clock runs asynchronous to the UART reference clock.

Reference Clock

The generation of the reference clock in the PS clock subsystem is controlled by the slcr.UART_CLK_CTRL register. This register can select the PLL that the clock is derived from and sets the divider frequency. This register also controls the clock enables for each UART controller. The generation of the UART reference clock is described in section [25.7.3 SDIO, SMC, SPI, Quad-SPI and UART Clocks](#).

Example: Configure Reference Clock

The clock can be based on any of the PLLs in the PS clock subsystem. In this example, the I/O PLL is used with a 1000 MHz clock. The clock divisor is 0x14 to generate a 50 MHz clock for the UART controllers.

1. **Program the UART Reference clock.** Write 0x0000_1401 to the slcr.UART_CLK_CTRL register.
 - a. Clock divisor, slcr.UART_CLK_CTRL[DIVISOR] = 0x14.
 - b. Select the IO PLL, slcr.UART_CLK_CTRL[SRCSEL] = 0.
 - c. Enable the UART 0 Reference clock, slcr.UART_CLK_CTRL[CLKACT0] = 1.
 - d. To enable UART 1 Reference clock, set the CLKACT1 bit = 1.

19.4.2 Resets

The controller reset bits are generated by the PS, see [Chapter 26, Reset System](#). These reset bits are not self-clearing and hence should be cleared to pull the peripheral out of reset.

Example: Software Reset

1. **Assert Controller Reset:** Set both slcr.UART_RST_CTRL[UARTx_REF_RST, UARTx_CPU1X_RST] bits = 1.
2. **De-assert Controller Reset:** Clear both slcr.UART_RST_CTRL[UARTx_REF_RST, UARTx_CPU1X_RST] bits = 0.

19.5 I/O Interface

19.5.1 MIO Programming

The UART Rx/D and Tx/D signals can be routed to one of many sets of MIO pins or to the EMIO interface. All of the modem flow control signals are always routed to the EMIO interface and are not available on the MIO pins. All of the UART signals are listed in [Table 19-4](#). The routing of the Rx/D and Tx/D signals are described in section [2.4 MIO-EMIO](#).

Example: Route UART 0 Rx/D/TxD Signals to MIO Pins 46, 47

In this example, the UART 0 Rx/D and Tx/D signals are routed through MIO pins 46 and 47. Many other pin options are possible.

- Configure MIO pin 46 for the Rx/D signal.** Write 0x0000_12E1 to the slcr.MIO_PIN_46 register:
 - Route UART 0 Rx/D signal to pin 46.
 - Output disabled (set TRI_ENABLE = 1).
 - LVC MOS18 (refer to the register definition for other voltage options).
 - Slow CMOS edge (benign setting).
 - Enable internal pull-up resistor.
 - Disable HSTL receiver.
- Configure MIO pin 47 for the Tx/D signal.** Write 0x0000_12E0 to the slcr.MIO_PIN_47 register:
 - Route UART 0 Tx/D signal to pin 47.
 - 3-state controlled by CAN (TRI_ENABLE = 0).
 - LVC MOS18 (refer to the register definition for other voltage options).
 - Slow CMOS drive edge.
 - Enable internal pull-up resistor.

Disable HSTL receiver.

19.5.2 MIO – EMIO Signals

The UART I/O signals are identified in [Table 19-4](#). The MIO pins and any restrictions based on device versions are shown in the MIO table in section [2.4.4 MIO-at-a-Glance Table](#).

Table 19-4: UART MIO Pins and EMIO Signals

UART Interface Signal	Default Controller Input Value	MIO Pins	I/O	EMIO Signals	
		Numbers		Name	I/O
UART 0 Transmit	~	11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51	O	EMIOUART0TX	O
UART 0 Receive		10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50	I	EMIOUART0RX	I

Table 19-4: UART MIO Pins and EMIO Signals (Cont'd)

UART Interface Signal	Default Controller Input Value	MIO Pins	I/O	EMIO Signals	
		Numbers		Name	I/O
UART 0 Clear to Send		~	~	EMIOUART0CTSN	I
UART 0 Ready to Send	~	~	~	EMIOUART0RTSN	O
UART 0 Data Set Ready		~	~	EMIOUART0DSRN	I
UART 0 Data Carrier Detect		~	~	EMIOUART0DCDN	I
UART 0 Ring Indicator		~	~	EMIOUART0RIN	I
UART 0 Data Terminal Ready	~	~	~	EMIOUART0DTRN	O
UART 1 Transmit	~	8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52	O	EMIOUART1TX	O
UART 1 Receive		9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53	I	EMIOUART1RX	I
UART 1 Clear to Send		~	~	EMIOUART1CTSN	I
UART 1 Ready to Send	~	~	~	EMIOUART1RTSN	O
UART 1 Data Set Ready		~	~	EMIOUART1DSRN	I
UART 1 Data Carrier Detect		~	~	EMIOUART1DCDN	I
UART 1 Ring Indicator		~	~	EMIOUART1RIN	I
UART 1 Data Terminal Ready	~	~	~	EMIOUART1DTRN	O

I2C Controller

20.1 Introduction

This I2C module is a bus controller that can function as a master or a slave in a multi-master design. It supports an extremely wide clock frequency range from DC (almost) up to 400 Kb/s.

In master mode, a transfer can only be initiated by the processor writing the slave address into the I2C register. The processor is notified of any available received data by a data interrupt or a transfer complete interrupt. If the HOLD bit is set, the I2C interface holds the SCL line Low after the data is transmitted to support slow processor service. The master can be programmed to use both normal (7-bit) addressing and extended (10-bit) addressing modes.

In slave monitor mode, the I2C interface is set up as a master and continues to attempt a transfer to a particular slave until the slave device responds with an ACK.

In slave mode, the extended address support is automatic by detecting a specific code in bits [7:3] of the first address byte. Extended addresses beyond seven bits are not supported. The HOLD bit can be set to prevent the master from continuing with the transfer, preventing an overflow condition in the slave.

A common feature between master mode and slave mode is the timeout (TO) interrupt flag. If at any point the SCL line is held Low by the master or the accessed slave for more than the period specified in the Timeout register, a timeout (TO) interrupt is generated to avoid stall conditions.

20.1.1 Features

The PS supports two I2C devices with these key features:

- I2C bus specification version 2
- Supports 16-byte FIFO
- Programmable normal and fast bus data rates
- Master mode
 - Write transfer
 - Read transfer
 - Extended address support
 - Support HOLD for slow processor service

- Supports TO interrupt flag to avoid stall condition
- Slave monitor mode
- Slave mode
 - Slave transmitter
 - Slave receiver
 - Extended address support (up to 7 address bits)
 - Fully programmable slave response address
 - Supports HOLD to prevent overflow condition
 - Supports TO interrupt flag to avoid stall condition
- Software can poll for status or function as interrupt-driven device
- Programmable interrupt generation

20.1.2 System Block Diagram

The system viewpoint diagram for the I2C module is shown in Figure 20-1.

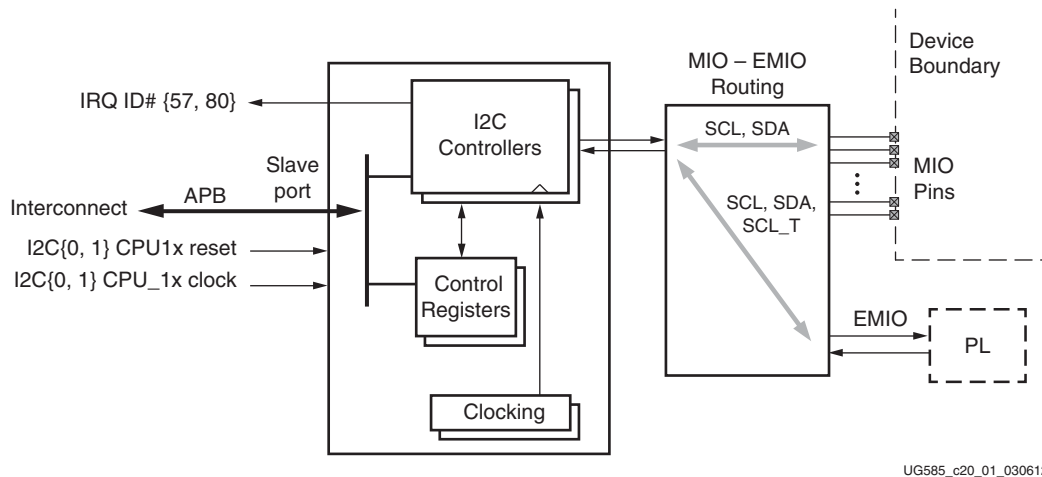


Figure 20-1: I2C System Block Diagram

20.2 Functional Description

20.2.1 Block Diagram

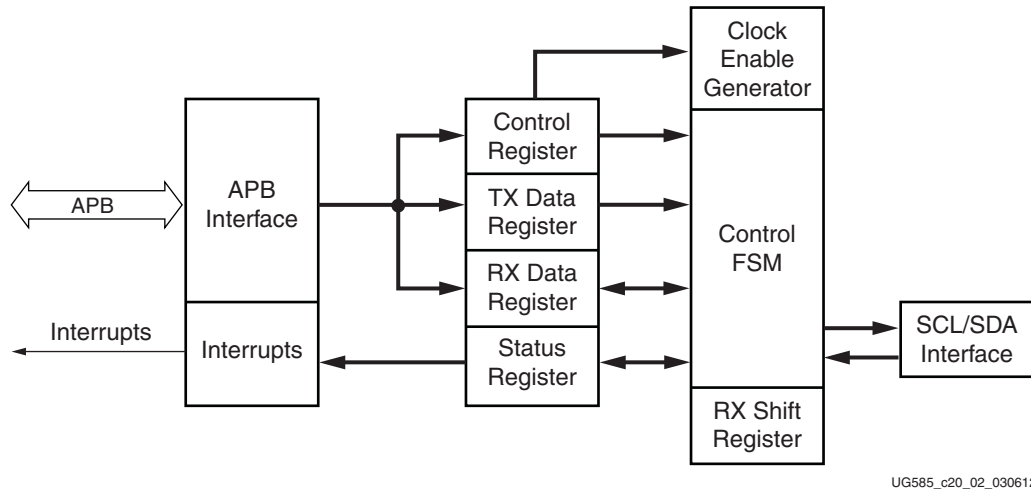


Figure 20-2: I2C Peripheral Block Diagram

20.2.2 Master Mode

An I2C transfer can only be initiated by the APB host, and two types of transfers can be performed:

- Write transfer, where the I2C becomes master transmitter
- Read transfer, where the I2C becomes master receiver

Write Transfer

To accomplish an I2C write transfer, the host must perform these steps:

1. Write to the control register to set up SCL speed and addressing mode.
2. Set the MS, ENACK, and CLR_FIFO bits and clear the RW bit.
3. If required, set the HOLD bit. Otherwise write the first byte of data to the I2C Data register.
4. Write the slave address into the I2C register. This initiates the I2C transfer.
5. Continue to load the remaining data to be sent to the slave by writing to the I2C Data register. The data is pushed in the FIFO each time the host writes to the I2C Data register.

When all data is transferred successfully, the COMP bit is set in the interrupt status register. A data interrupt is generated whenever there are only two bytes left for transmission in the FIFO.

If the HOLD bit is not set when the data is transmitted, the I2C interface generates a STOP condition and terminates the transfer. If the HOLD bit is set, the I2C interface holds the SCL line Low after the

data is transmitted. The host is notified of this event by a transfer complete interrupt (COMP bit set) and the TXDV bit in the status register is cleared. At this point, the host can proceed in three ways:

1. Clear the HOLD bit. This causes the I2C interface to generate a STOP condition.
2. Supply more data by writing to the I2C register. This causes the I2C interface to continue with the transfer, writing more data to the slave.
3. Perform combined format transfer. This is accomplished by first writing to the Control register and, if necessary, changing the transfer direction or addressing mode. The host must then write to the I2C Address register, which leads to a RESTART condition being generated by the I2C interface.

If at any point the slave responds with a NACK, the transfer automatically terminates and a transfer NACK interrupt is generated (the NACK bit set). The outstanding amount of data minus one is then read from the Transfer Size register. Unless the very last byte written by the host into the FIFO was a NACK byte, TXDV remains High. In this case, the host must clear the FIFO by setting the CLR_FIFO bit in the Control register.

If at any point the SCL line is held Low by the master or the accessed slave for more than the period specified in the Timeout register, a TO interrupt is generated and the outstanding amount of data minus one is then read from the Transfer Size register.

Read Transfer

To accomplish an I2C read transfer, the host must perform these steps:

1. Write to the Control register to set up the SCL speed and addressing mode.
2. Set MS, ENACK, CLR_FIFO bits, and RW bit.
3. If the host wants to hold the bus after the data is received, it must also set the HOLD bit.
4. Write the slave address in the I2C Address register. This initiates the I2C transfer.
5. Write the number of requested bytes in the Transfer Size register.

The host is notified of any available received data in two ways:

- If an outstanding transfer size is more than 126 bits, a data interrupt is generated (DATA bit is set).
- If an outstanding transfer size is less than 126 bits, a transfer complete interrupt is generated (COMP bit is set).

In both cases, the RXDV bit in the status register is set.

The I2C interface automatically returns a NACK after receiving the last expected byte and terminates the transfer by generating a STOP condition. If the HOLD bit is set during a master read transfer, the I2C interface drives the SCL line Low.

If at any point the slave responds with NACK while the master transmits a slave address for a master read transfer, the transfer automatically terminates and a transfer NACK interrupt is generated (NACK bit is set). The outstanding amount of data is read from the Transfer Size register.

If at any point the SCL line is held Low by the master or the accessed slave for more than the period specified in the Timeout register, a TO interrupt is generated.

The I2C module, as a master, is capable of performing combined transfers as specified by the I2C specification. It can perform more than one transfer without releasing the I2C bus and these transfers are separated by a repeated START condition.

The host can realize combined transfers by writing to the Address register before the end of the previous transfer. The correct transfer direction and slave addressing mode must be set by writing to the Control register, and then the Address register can be accessed for the second part of a combined transfer.

For combined transfers, the host must set the HOLD bit in the beginning of the first part of a transfer. When the host receives a transfer complete interrupt, it can then initiate the second part of a combined transfer.

20.2.3 Slave Monitor Mode

In slave monitor mode, the I2C interface is set up as a master. The host must set the MS and SLVMON bits and clear the RW bit in the Control register. Also, it must initialize the Slave Monitor Pause register.

The master attempts a transfer to a particular slave whenever the host writes to the I2C Address register. If the slave returns a NACK when it receives the address, the master waits for the time interval established by the Slave Monitor Pause register and attempts to address the slave again. The master continues this cycle until the slave responds with an ACK to its address or until the host clears the SLVMON bit in the Control register. If the addressed slave responds with an ACK, the I2C interface terminates the transfer by generating a STOP condition and a SLV_RDY interrupt.

20.2.4 Slave Mode

The I2C interface is set up as a slave by clearing the MS bit in the Control register. The I2C slave must be given a unique identifying address by writing to the I2C address register. The SCL speed must also be set up at least as fast as the fastest SCL frequency expected to be seen. When in slave mode, the I2C interface operates as either a slave transmitter or a slave receiver.

Extended Address Support

The I2C interface supports a slave address with up to seven bits. Extended addresses beyond seven bits are not supported.

Slave Transmitter

The slave becomes a transmitter after recognizing the entire slave address sent by the master and when the R/W field in the last address byte sent is High. This means that the slave has been requested to send data over the I2C bus and the host is notified of this through an interrupt. At the same time, the SCL line is held Low to allow the host to supply data to the I2C slave before the I2C master starts sampling the SDA line. The host is notified of this event by setting the DATA interrupt flag.

At the same time, the SCL line is held Low to allow the host to supply data to the I2C slave before the I2C master starts sampling the SDA line.

The host must supply data for transmission through the I2C data register so that the SCL line is released and transfer continues. If it does not write to the I2C data register before the timeout period expires, an interrupt is generated and a TO interrupt flag is set.

After the host writes to the I2C data register, the transfer continues by loading data in the FIFO while the transfer is in progress. The amount of data loaded in the FIFO might be a known system parameter or communicated in advance through a higher level protocol using the I2C bus.

When there are only two valid bytes left in the FIFO for transmission, an interrupt is generated and the DATA interrupt flag is set. If the I2C master returns a NACK on the last byte transmitted from the FIFO, an interrupt is generated, and the COMP interrupt flag is set as soon as the I2C master generates a STOP condition.

Transfer must continue if the master acknowledges on the last byte sent out from the FIFO. At that moment, the DATA interrupt flag is set. The TXDV flag in the Status register is cleared because the FIFO is empty.

If the I2C master terminates the transfer before all of the data in the FIFO is sent by the slave, the host is notified, and the NACK interrupt flag is set while TXDV remains set and the Transfer Size register indicates the remaining bytes in the FIFO. The host must set the CLR_FIFO bit in the Control register to clear the FIFO and the TXDV bit.

Slave Receiver

The slave becomes a receiver after recognizing the entire slave address sent by the master and when the R/W bit in the first address byte is Low. This means that the master is about to send one or more data bytes to the slave over the I2C bus.

An interrupt is generated and the DATA interrupt flag is set when there are only two free locations left in the FIFO. After a byte is acknowledged by the I2C slave, the RXDV bit in the Status register is set, indicating that new data has been received. The host reads the received data through the I2C Data register.

Whenever the I2C master generates a STOP condition, an interrupt is generated and the COMP interrupt flag is set. The Transfer Size register then contains the number of bytes received that are available in the FIFO. This number is decremented by the host on each read of the I2C Data register.

If the FIFO is full when one or more bytes are received by the I2C interface, an interrupt is generated and the RX_OVF interrupt flag is set. The last byte received is not acknowledged and the data in the FIFO is kept intact.

The HOLD bit can be set in the Control register to avoid overflow conditions when it is impossible to respond to interrupts in a reasonable time. If the HOLD bit is set, the I2C interface keeps the SCL line Low until the host clears resources for data reception. This prevents the master from continuing with the transfer, causing an overflow condition in the slave. The host clears resources for data reception by reading the data register.

If the HOLD bit is set and the I2C interface keeps the SCL line Low for longer than the timeout period, an interrupt is generated and the TO interrupt flag is set.

20.2.5 I2C Speed

The main clock used within the I2C interface is the clock enable signal (see [Figure 20-3](#)).

- In slave mode, the clock enable is used to extract synchronization information for correct sampling of the SDA line.
- In master mode, the clock enable is used to establish a time base for generating the desired SCL frequency.

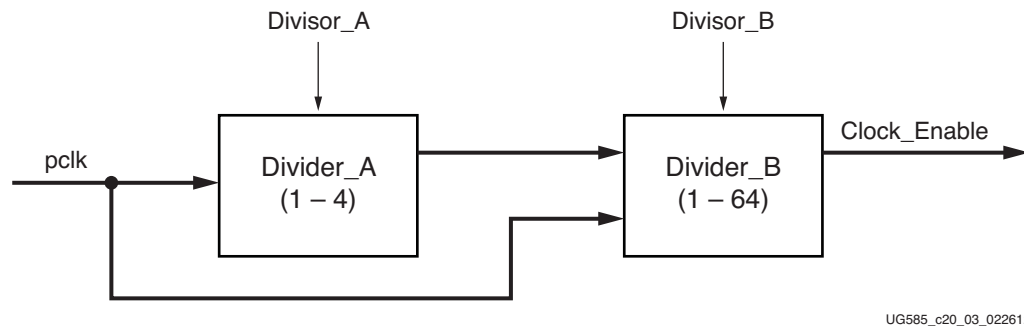


Figure 20-3: I2C Clock Generator

The frequency of the SCL output is exactly 22 times slower than the frequency of clock_enable. The frequency of the clock_enable signal is defined by the frequency of the cpu_1xclk (pclk) and the values of divisor_a and divisor_b using [Equation 20-1](#)

$$F_{clock_enable} = \frac{F_{pclk}}{(divisor_a + 1) \times (divisor_b + 1)} \quad \text{Equation 20-1}$$

See I2C register map in [Appendix B, Register Details](#) for details of register fields.

20.2.6 Multi-Master Operation

In I2C multi master mode, the Zynq-7000 EPP device acts as a master sharing the bus with other masters. In this mode, the I2C clock (I2C_SCL) is driven by the device that acts as the master.

20.3 Register Overview

An overview of the I2C registers is provided in [Table 20-1](#).

Table 20-1: I2C Register Overview

Function	Register Names	Overview
Configuration	Control_reg0	Configure the operating mode
Data	I2C_address_reg0 I2C_data_reg0 Transfer_size_register0 Slave_mon_pause_reg0 Time_out_reg0 Staus_reg0	Transfer data and monitors status.
Interrupt processing	Intrpt_status_reg0 Intrpt_mask_reg0 Intrpt_enable_reg0 Intrpt_disable_reg0	Enable/disable interrupt detection, mask interrupt set to the interrupt controller, read raw interrupt status.

20.4 Interface Signals

[Table 20-2](#) identifies the interface signals to the I2C controller. The MIO pins and any restrictions based on device version are shown in the MIO table in section [2.4.4 MIO-at-a-Glance Table](#).

Table 20-2: I2C MIO Pins and EMIO Signals

I2C Interface	Default Controller Input Value	MIO Pins	I/O	EMIO Signals	
		Numbers		Name	I/O
I2C 0, Serial Clock	0	10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50	IO	EMIOI2C0SCLI	O
				EMIOI2C0SCLO	I
				EMIOI2C0SCLTN	I
I2C 0, Serial Data	~	11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51	IO	EMIOI2C0SDAI	O
				EMIOI2C0SDAO	I
				EMIOI2C0SDATN	I
I2C 1, Serial Clock	0	12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52	IO	EMIOI2C1SCLI	O
				EMIOI2C1SCLO	I
				EMIOI2C1SCLTN	I
I2C 1, Serial Data	~	13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53	IO	EMIOI2C1SDAI	O
				EMIOI2C1SDAO	I
				EMIOI2C1SDATN	I

Programmable Logic Description

21.1 Introduction

The Zynq-7000 devices integrates a feature-rich dual-core ARM® Cortex™-A9 MPCore™ based processing system (PS) and Xilinx programmable logic (PL) in a single device. Each Zynq-7000 EPP device contains the same PS while the PL and I/O resources vary between the devices. The PL of the two smaller EPP devices, Z-7010 and Z-7020 is based on Artix™-7 FPGA logic while the two bigger devices, Z-7030 and Z-7045 are based on Kintex™-7 FPGA logic. For details about resources, see DS190, *Zynq-7000 Extensible Processing Platform Overview*.

The PS and PL can be tightly or loosely coupled using multiple interfaces and other signals that have a combined total of over 3,000 connections. This enables the designer to effectively integrate user-created hardware accelerators and other functions in the PL logic that are accessible to the processors and can also access memory resources in the PS. Zynq customers are able to differentiate their product in hardware by customizing their applications using PL.

The processors in the PS always boot first, allowing a software centric approach for PL configuration. The PL can be configured as part of the boot process or configured at some point in the future. Additionally, the PL can be completely reconfigured or used with partial, dynamic reconfiguration (PR). PR allows configuration of a portion of the PL. This enables optional design changes such as updating coefficients or time-multiplex the PL resources by swapping in new algorithms as needed. This latter capability is analogous to the dynamic loading and unloading of soft-ware modules. The PL configuration data is referred to as a bitstream.

The PL can be on a separate power domain from the PS. This enables users to save power by completely shutting down the PL. In this mode, the PL consumes no static or dynamic power thus significantly reducing the power consumption of the device. The PL must be reconfigured when coming out of this mode. Users need to take into account the re-configuration time of the PL for their particular application as this varies depending on the size of the bitstream for their application.

21.1.1 Features

The PL provides a rich architecture of user-configurable capabilities. The key features are:

- Configurable logic blocks (CLB)
 - 6-input look-up tables (LUTs)
 - Memory capability within the LUT
 - Register and shift register functionality

- Cascadeable adders
- 36 Kb block RAM
 - Dual port
 - Up to 72-bits wide
 - Configurable as dual 18 Kb
 - Programmable FIFO logic
 - Built-in error correction circuitry
- Digital signal processing – DSP48E1 Slice
 - 25 × 18 two's complement multiplier/accumulator high-resolution (48 bit) signal processor
 - Power saving 25-bit pre-adder to optimize symmetrical filter applications
 - Advanced features: optional pipelining, optional ALU, and dedicated buses for cascading
- Clock management
 - High-speed buffers and routing for low-skew clock distribution
 - Frequency synthesis and phase shifting
 - Low-jitter clock generation and jitter filtering
- Configurable I/Os
 - High-performance SelectIO technology
 - High-frequency decoupling capacitors within the package for enhanced signal integrity
 - Digitally controlled impedance that can be 3-stated for lowest power, high-speed I/O operation
 - High range (HR) IOs support 1.2V to 3.3V
 - High performance (HP) IOs support 1.2V to 1.8V (Z-7030 and Z-7045 devices)
- Low-power serial transceivers (Z-7030 and Z-7045 devices)
 - High-performance transceivers capable of up to 12.5 Gb/s (GTX)
 - Low-power mode optimized for chip-to-chip interfaces
 - Advanced transmit pre and post emphasis, and receiver linear (CTLE) and decision feedback equalization (DFE), including adaptive equalization for additional margin
- XADC (analog-to-digital converter)
 - Dual 12-bit 1 MSPS analog-to-digital converters (ADCs)
 - Up to 17 flexible and user-configurable analog inputs
 - On-chip or external reference option
 - On-chip temperature ($\pm 4^{\circ}\text{C}$ max error) and power supply ($\pm 1\%$ max error) sensors
 - Continuous JTAG access to ADC measurements
- Integrated interface block for PCI Express designs
 - Compatible with the PCI Express Base Specification 2.1 with Endpoint and Root Port capability
 - Supports Gen2 (5.0 Gb/s)

- Advanced configuration options, advanced error reporting (AER), and end-to-end CRC (ECRC) advanced error reporting and ECRC features

21.2 Detailed Description

21.2.1 CLBs, Slices, and LUTs

Some key features of the Configurable Logic Blocks (CLB) architecture include:

- True 6-input Look Up Tables (LUT)s
- Memory capability within the LUT
- Register and shift register functionality

The LUTs in the Zynq-7000 EPP can be configured as either one 6-input LUT (64-bit ROMs) with one output, or as two 5-input LUTs (32-bit ROMs) with separate outputs but common addresses or logic inputs. Each LUT output can optionally be registered in a flip-flop. Four such LUTs and their eight flip-flops as well as multiplexers and arithmetic carry logic form a slice, and two slices form a configurable logic block (CLB). One flip-flop per LUT can optionally be configured as latches.

Between 25–50% of all slices can also use their LUTs as distributed 64-bit RAM or as 32-bit shift registers (SRL32) or as two SRL16s. Modern synthesis tools take advantage of these highly efficient logic, arithmetic, and memory features.

For more details on Configuration Logic Blocks, see [UG474](#), *7 Series FPGAs Configurable Logic Block User Guide*.

21.2.2 Clock Management

Some of the key highlights of the clock management architecture include:

- High-speed buffers and routing for low-skew clock distribution
- Frequency synthesis and phase shifting
- Low-jitter clock generation and jitter filtering

Each Zynq-7000 EPP device has up to eight clock management tiles (CMTs), each consisting of one mixed-mode clock manager (MMCM) and one phase-locked loop (PLL).

Mixed-Mode Clock Manager and Phase-Locked Loop

The mixed-mode clock manager (MMCM) and the phase-locked loop (PLL) share many characteristics. Both can serve as a frequency synthesizer for a wide range of frequencies and as a jitter filter for incoming clocks. At the center of both components is a voltage-controlled oscillator (VCO), which speeds up and slows down depending on the input voltage it receives from the phase frequency detector (PFD).

There are three sets of programmable frequency dividers: D, M, and O. The pre-divider D (programmable by configuration and afterwards via Dynamic Configuration Port (DRP)) reduces the input frequency and feeds one input of the traditional PLL phase/frequency comparator. The feedback divider M (programmable by configuration and afterwards via DRP) acts as a multiplier because it divides the VCO output frequency before feeding the other input of the phase comparator. The values of D and M must be chosen appropriately to keep the VCO within its specified frequency range.

The VCO has eight equally-spaced output phases (0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°). Each can be selected to drive one of the output dividers (six for the PLL, O0 to O5, and seven for the MMCM, O0 to O6), each programmable by configuration to divide by any integer from 1 to 128.

The MMCM and PLL have three input-jitter filter options: Low-bandwidth mode which has the best jitter attenuation; high-bandwidth mode, which has the best phase offset; and optimized mode, which allows the tools to find the best setting.

MMCM Additional Programmable Features

The MMCM can have a fractional counter in either the feedback path (acting as a multiplier) or in one output path. Fractional counters allow non-integer increments of 1/8 and can thus increase frequency synthesis capabilities by a factor of eight.

The MMCM can also provide fixed or dynamic phase shift in small increments that depend on the VCO frequency. At 1,600 MHz, the phase-shift timing increment is 11.2 ps.

Clock Distribution

Each Zynq-7000 EPP device provides six different types of clock lines (BUFG, BUFR, BUFIO, BUFH, BUFMR, and the high-performance clock) to address the different clocking requirements of high fanout, short propagation delay, and extremely low skew.

Global Clock Lines

In each Zynq-7000 EPP device, 32 global clock lines have the highest fanout and can reach every flip-flop clock, clock enable, and set/reset as well as many logic inputs. There are 12 global clock lines within any clock region driven by the horizontal clock buffers (BUFH). Each BUFH can be independently enabled/disabled, allowing for clocks to be turned off within a region, thereby offering fine-grain control over which clock regions consume power. Global clock lines can be driven by global clock buffers, which can also perform glitchless clock multiplexing and clock enable functions. Global clocks are often driven from the CMT, which can completely eliminate the basic clock distribution delay.

Regional Clocks

Regional clocks can drive all clock destinations in their region. A region is defined as any area that is 50 I/O and 50 CLB high and half the device wide. Zynq-7000 EPP devices have between eight and twenty-four regions. There are four regional clock tracks in every region. Each regional clock buffer can be driven from either of four clock-capable input pins, and its frequency can optionally be divided by any integer from 1 to 8.

I/O Clocks

I/O clocks are especially fast and serve only I/O logic and serializer/deserializer (SerDes) circuits, as described in [Input/Output](#).

Zynq-7000 EPP devices have a direct connection from the MMCM to the I/O for low-jitter, high-performance interfaces.

For more details on clocking resources, see [UG472, Series FPGAs Clocking Resources User Guide](#).

21.2.3 Block RAM

Some of the key features of the block RAM include:

- Dual-port 36 Kb block RAM with port widths of up to 72
- Programmable FIFO logic
- Built-in optional error correction circuitry

Every Zynq-7000 EPP device has between 60 and 465 dual-port block RAMs, each storing 36 Kb. Each block RAM has two completely independent ports that share nothing but the stored data.

Synchronous Operation

Each memory access, read or write, is controlled by the clock. All inputs, data, address, clock enables, and write enables are registered. The input address is always clocked, retaining data until the next operation. An optional output data pipeline register allows higher clock rates at the cost of an extra cycle of latency.

During a write operation, the data output can reflect either the previously stored data, the newly written data, or can remain unchanged.

Programmable Data Width

Each port can be configured as $32K \times 1$, $16K \times 2$, $8K \times 4$, $4K \times 9$ (or 8), $2K \times 18$ (or 16), $1K \times 36$ (or 32), or 512×72 (or 64). The two ports can have different widths without any constraints.

Each block RAM can be divided into two completely independent 18 Kb block RAMs that can each be configured to any aspect ratio from $16K \times 1$ to 512×36 . Everything described previously for the full 36 Kb block RAM also applies to each of the smaller 18 Kb block RAMs.

Only in simple dual-port (SDP) mode can data widths of greater than 18 bits (18 Kb RAM) or 36 bits (36 Kb RAM) be accessed. In this mode, one port is dedicated to read operations, the other to write operations. In SDP mode, one side (read or write) can be variable, while the other is fixed to 32/36 or 64/72.

Both sides of the dual-port 36 Kb RAM can be of variable width.

Two adjacent 36 Kb block RAMs can be configured as one $64K \times 1$ dual-port RAM without any additional logic.

Error Detection and Correction

Each 64-bit-wide block RAM can generate, store, and utilize eight additional Hamming code bits and perform single-bit error correction and double-bit error detection (ECC) during the read process. The ECC logic can also be used when writing to or reading from external 64- to 72-bit-wide memories.

FIFO Controller

The built-in FIFO controller for single-clock (synchronous) or dual-clock (asynchronous or multirate) operation increments the internal addresses and provides four handshaking flags: full, empty, almost full, and almost empty. The almost full and almost empty flags are freely programmable. Similar to the block RAM, the FIFO width and depth are programmable, but the write and read ports always have identical width.

First word fall-through mode presents the first-written word on the data output even before the first read operation. After the first word has been read, there is no difference between this mode and the standard mode.

For more details on Block RAM, see [UG473](#), *7 Series FPGAs Memory Resources User Guide*.

21.2.4 Digital Signal Processing — DSP Slice

Some highlights of the DSP functionality include:

- 25 × 18 two's complement multiplier/accumulator high-resolution (48 bit) signal processor
- Power saving pre-adder to optimize symmetrical filter applications
- Advanced features: optional pipelining, optional ALU, and dedicated buses for cascading

DSP applications use many binary multipliers and accumulators, best implemented in dedicated DSP slices. All Zynq-7000 EPP devices have many dedicated, full custom, low-power DSP slices, combining high speed with small size while retaining system design flexibility.

Each DSP slice fundamentally consists of a dedicated 25 × 18 bit two's complement multiplier and a 48-bit accumulator, both capable of operating up to 741 MHz. The multiplier can be dynamically bypassed, and two 48-bit inputs can feed a single-instruction-multiple-data (SIMD) arithmetic unit (dual 24-bit add/subtract/accumulate or quad 12-bit add/subtract/accumulate), or a logic unit that can generate any one of ten different logic functions of the two operands.

The DSP includes an additional pre-adder, typically used in symmetrical filters. This pre-adder improves performance in densely packed designs and reduces the DSP slice count by up to 50%. The DSP also includes a 48-bit-wide pattern detector that can be used for convergent or symmetric rounding. The pattern detector is also capable of implementing 96-bit-wide logic functions when used in conjunction with the logic unit.

The DSP slice provides extensive pipelining and extension capabilities that enhance the speed and efficiency of many applications beyond digital signal processing, such as wide dynamic bus shifters, memory address generators, wide bus multiplexers, and memory-mapped I/O register files. The accumulator can also be used as a synchronous up/down counter.

For more details on DSP slices, see [UG479](#), *7 Series FPGAs DSP48E1 User Guide*.

21.2.5 Input/Output

Some highlights of the input/output functionality include:

- High-performance SelectIO technology with support for 1,866 Mb/s DDR3
- High-frequency decoupling capacitors within the package for enhanced signal integrity
- Digitally controlled impedance that can be 3-stated for lowest power, high-speed I/O operation

The number of I/O pins varies depending on device and package size. Each I/O is configurable and can comply with a large number of I/O standards. With the exception of the supply pins and a few dedicated configuration pins, all other PL pins have the same I/O capabilities, constrained only by certain banking rules. The SelectIO resources in Zynq-7000 EPP devices are classed as either high range (HR) or high performance (HP). The HR I/Os offer the widest range of voltage support, from 1.2V to 3.3V. The HP I/Os are optimized for highest performance operation, from 1.2V to 1.8V.

All I/O pins are organized in banks, with 50 pins per bank. Each bank has one common V_{CCO} output supply, which also powers certain input buffers. Some single-ended input buffers require an internally generated or an externally applied reference voltage (V_{REF}). There are two V_{REF} pins per bank (except configuration bank 0). A single bank can have only one V_{REF} voltage value.

Zynq-7000 EPP devices use a variety of package types to suit the needs of the user, including small form factor wire-bond packages for lowest cost; conventional, high performance flip-chip packages; and lidless flip-chip packages that balance smaller form factor with high performance. In the flip-chip packages, the silicon device is attached to the package substrate using a high-performance flip-chip process. Controlled ESR discrete decoupling capacitors are mounted on the package substrate to optimize signal integrity under simultaneous switching of outputs (SSO) conditions.

I/O Electrical Characteristics

Single-ended outputs use a conventional CMOS push/pull output structure driving High towards V_{CCO} or Low towards ground, and can be put into a high-Z state. The system designer can specify the slew rate and the output strength. The input is always active but is usually ignored while the output is active. Each pin can optionally have a weak pull-up or a weak pull-down resistor.

Most signal pin pairs can be configured as differential input pairs or output pairs. Differential input pin pairs can optionally be terminated with a 100 Ω internal resistor. All Zynq-7000 EPP devices support differential standards beyond LVDS: HT, RSDS, BLVDS, differential SSTL, and differential HSTL.

Each of the I/Os supports memory I/O standards, such as single-ended and differential HSTL as well as single-ended SSTL and differential SSTL. The SSTL I/O standard can support data rates of up to 1,866 Mb/s for DDR3 interfacing applications.

3-State Digitally Controlled Impedance and Low-Power I/O Features

The 3-state digitally controlled impedance (T_DCI) can control the output drive impedance (series termination) or can provide parallel termination of an input signal to V_{CCO} or split (Thevenin) termination to $V_{CCO}/2$. This allows users to eliminate off-chip termination for signals using T_DCI. In addition to board space savings, the termination automatically turns off when in output mode or when 3-stated, saving considerable power compared to off-chip termination. The I/Os also have

low-power modes for IBUF and IDELAY to provide further power savings, especially when used to implement memory interfaces.

I/O Logic

Input and Output Delay

All inputs and outputs can be configured as either combinatorial or registered. Double data rate (DDR) is supported by all inputs and outputs. Any input and some outputs can be individually delayed by up to 32 increments of 78 ps or 52 ps each.

Such delays are implemented as IDELAY and ODELAY. The number of delay steps can be set by configuration and can also be incremented or decremented while in use. ODELAY is only available for HP Select I/O. It is not available for HR select I/Os. This means that it is only available for Z-7030 and Z-7045 devices.

ISERDES and OSERDES

Many applications combine high-speed, bit-serial I/O with slower parallel operation inside the device. This requires a serializer and deserializer (SerDes) inside the I/O structure. Each I/O pin possesses an 8-bit IOSERDES (ISERDES and OSERDES) capable of performing serial-to-parallel or parallel-to-serial conversions with programmable widths of 2, 3, 4, 5, 6, 7, or 8 bits. By cascading two IOSERDES from two adjacent pins (default from differential I/O), wider width conversions of 10 and 14 bits can also be supported.

The ISERDES has a special oversampling mode capable of asynchronous data recovery for applications like a 1.25 Gb/s LVDS I/O-based SGMII interface.

For more details on Select I/Os, see [UG471](#), *7 Series FPGAs SelectIO Resources User Guide*.

21.2.6 Low-Power Serial Transceivers

Low-power serial transceivers are only available in Z-7030 and Z-7045 devices. Z-7030 has 4 transceivers and Z-7045 has 16. Some highlights of the low-power gigabit transceivers include:

- High-performance transceivers capable of up to 12.5 Gb/s line rates with flipchip packages and up to 6.6Gb/s with bare-die flipchip packages.
- Low-power mode optimized for chip-to-chip interfaces.
- Advanced Transmit pre and post emphasis, and receiver linear (CTLE) and decision feedback equalization (DFE), including adaptive equalization for additional margin

Ultra-fast serial data transmission to optical modules, between ICs on the same PCB, over the backplane, or over longer distances is becoming increasingly popular and important to enable customer line cards to scale to 200 Gb/s. It requires specialized dedicated on-chip circuitry and differential I/O capable of coping with the signal integrity issues at these high data rates.

The Zynq-7000 EPP devices transceiver counts range from 0 to 16 transceiver circuits. Each serial transceiver is a combined transmitter and receiver. The various Zynq-7000 serial transceivers can use a combination of ring oscillators and LC tank architecture to allow the ideal blend of flexibility and

performance while enabling IP portability across the family members. Lower data rates can be achieved using logic-based oversampling of PL. The serial transmitter and receiver are independent circuits that use an advanced PLL architecture to multiply the reference frequency input by certain programmable numbers between 4 and 25 to become the bit-serial data clock. Each transceiver has a large number of user-definable features and parameters. All of these can be defined during device configuration, and many can also be modified during operation.

Transmitter

The transmitter is fundamentally a parallel-to-serial converter with a conversion ratio of 16, 20, 32, 40, 64, or 80. This allows the designer to trade-off datapath width for timing margin in high-performance designs. These transmitter outputs drive the PC board with a single-channel differential output signal. TXOUTCLK is the appropriately divided serial data clock and can be used directly to register the parallel data coming from the internal logic. The incoming parallel data is fed through an optional FIFO and has additional hardware support for the 8B/10B, 64B/66B, or 64B/67B encoding schemes to provide a sufficient number of transitions. The bit-serial output signal drives two package pins with differential signals. This output signal pair has programmable signal swing as well as programmable pre- and post-emphasis to compensate for PC board losses and other interconnect characteristics. For shorter channels, the swing can be reduced to reduce power consumption.

Receiver

The receiver is fundamentally a serial-to-parallel converter, changing the incoming bit-serial differential signal into a parallel stream of words, each 16, 20, 32, 40, 64, or 80 bits. This allows the designers to trade-off internal datapath width versus logic timing margin. The receiver takes the incoming differential data stream, feeds it through programmable linear and decision feedback equalizers (to compensate for PC board and other interconnect characteristics), and uses the reference clock input to initiate clock recognition. There is no need for a separate clock line. The data pattern uses non-return-to-zero (NRZ) encoding and optionally guarantees sufficient data transitions by using the selected encoding scheme. Parallel data is then transferred into the PL using the RXUSRCLK clock. For short channels, the transceivers offers a special low power mode (LPM) for additional power reduction.

Out-of-Band Signaling

The transceivers provide out-of-band (OOB) signaling, often used to send low-speed signals from the transmitter to the receiver while high-speed serial data transmission is not active. This is typically done when the link is in a powered-down state or has not yet been initialized. This benefits PCI Express and SATA/SAS applications.

For more details on GTX Transceivers, see [UG476](#), *7 Series FPGAs GTX Transceiver User Guide*.

21.2.7 Integrated Block for PCI Express Designs

The integrated PCI Express block is only available in Z-7030 and Z-7045 devices. Highlights of the integrated blocks for PCI Express include:

- Compatible with the PCI Express Base Specification 2.1 with Endpoint and Root Port capability
- Supports Gen1 (2.5 Gb/s) and Gen2 (5 Gb/s)
- Advanced configuration options, advanced error reporting (AER), and end-to-end CRC (ECRC) advanced error reporting and ECRC features in the integrated block depending on family

All Zynq-7000 EPP devices with transceivers include an integrated block for PCI Express technology that can be configured as an Endpoint or Root Port, compatible with the PCI Express Base Specification Revision 2.1. The Root Port can be used to build the basis for a compatible Root Complex, to allow custom communication between the Zynq-7000 EPP device and other devices via the PCI Express protocol, and to attach ASSP Endpoint devices, such as Ethernet controllers or fibre channel HBAs, to the Zynq-7000 devices.

This block is highly configurable to system design requirements and can operate 1, 2, 4, or 8 lanes at the 2.5 Gb/s and 5.0 Gb/s data rates. For high-performance applications, advanced buffering techniques of the block offer a flexible maximum payload size of up to 1,024 bytes. The integrated block interfaces to the integrated high-speed transceivers for serial connectivity and to block RAMs for data buffering. Combined, these elements implement the physical layer, data link layer, and transaction layer of the PCI Express protocol.

Xilinx provides a light-weight, configurable, easy-to-use LogiCORE™ IP wrapper that ties the various building blocks (the integrated block for PCI Express, the transceivers, block RAM, and clocking resources) into an Endpoint or Root Port solution. The system designer has control over many configurable parameters: lane width, maximum payload size, programmable logic interface speeds, reference clock frequency, and base address register decoding and filtering.

Xilinx offers AXI4 memory mapped wrapper for the integrated block. AXI4 (memory mapped) is designed for Xilinx Platform Studio/EDK design flow and MicroBlaze™ processor based designs.

For more details on PCIe, see [UG477](#), *7 Series FPGAs Integrated Block v1.3 for PCI Express User Guide*.

21.2.8 XADC (Analog-to-Digital Converter)

Highlights of the XADC architecture include:

- Dual 12-bit 1MSPS analog-to-digital converters (ADCs)
- Up to 17 flexible and user-configurable analog inputs
- On-chip or external reference option
- On-chip temperature ($\pm 4^{\circ}\text{C}$ max error) and power supply ($\pm 1\%$ max error) sensors
- Continuous JTAG access to ADC measurements

All Zynq-7000 EPP devices integrate a new flexible analog interface called XADC. When combined with the programmable logic capability of the Zynq-7000 EPP devices, the XADC can address a broad range of data acquisition and monitoring requirements. This unique combination of analog and programmable logic is called Agile Mixed Signal. For more information, go to:

<http://www.xilinx.com/ams>.

The XADC contains two 12-bit 1MSPS ADCs with separate track and hold amplifiers, an on-chip analog multiplexer (up to 17 external analog input channels supported), and on-chip thermal and supply sensors. The two ADCs can be configured to simultaneously sample two external-input

analog channels. The track and hold amplifiers support a range of analog input signal types, including unipolar, bipolar, and differential. The analog inputs can support signal bandwidths of at least 500 KHz at sample rates of 1 MSPS. It is possible to support higher analog bandwidths using external analog multiplexer mode with the dedicated analog input (see [UG480](#), *7 Series FPGAs XADC Dual 12-Bit 1MSPS Analog-to-Digital Converter User Guide*).

The XADC optionally uses an on-chip reference circuit ($\pm 1\%$), thereby eliminating the need for any external active components for basic on-chip monitoring of temperature and power supply rails. To achieve the full 12-bit performance of the ADCs, an external 1.25V reference IC is recommended.

If the XADC is not instantiated in a design, then by default it digitizes the output of all on-chip sensors. The most recent measurement results (together with maximum and minimum readings) are stored in dedicated registers for access at any time via the JTAG interface. User-defined alarm thresholds can automatically indicate over-temperature events and unacceptable power supply variation. A user-specified limit (for example, 100°C) can be used to initiate an automatic power-down.

For more details on XADC, see [UG480](#), *7 Series FPGAs XADC Dual 12-Bit 1MSPS Analog-to-Digital Converter User Guide*.

21.2.9 Configuration

Xilinx 7 series FPGAs store their customized configuration in SRAM-type internal latches. The number of configuration bits is between 17 Mb and 102 Mb, depending on device size and user-design implementation options. The configuration storage is volatile and must be reloaded whenever the FPGA is powered up. The Processing System can reload the configuration at any time. For details about different boot and configuration modes refer to [Chapter 6, Boot and Configuration](#).

Table 21-1: Zynq Internal SRAM Latches Configurations

Zynq 7000 EPP Device	Length (Mb)	Length (MBs)
Z-7010	17	2.125
Z-7020	32.5	4.2
Z-7030	48	6
Z-7045	101.6	12.7

In all Zynq-7000 EPP devices, the PL bitstream, which contains sensitive customer IP, can be protected with 256-bit AES encryption and HMAC/SHA-256 authentication to prevent unauthorized copying of the design. The PL performs decryption on the fly during configuration using an internally stored 256-bit key. This key can reside in battery-backed RAM or in nonvolatile eFUSE bits.

Most configuration data can be read back without affecting the system's operation. Typically, configuration is an all-or-nothing operation, but Zynq-7000 EPP devices also support partial reconfiguration. This is an extremely powerful and flexible feature that allows the user to change portions of the logic in the PL while other portions remain static. Users can time-slice these portions to fit more IP into smaller devices, saving cost and power. Where applicable in certain designs, partial reconfiguration can greatly improve the versatility of the Zynq-7000 EPP device.

21.3 PS-PL Interfaces

The PS-PL interface contains all the signals available to the PL designer for integrating the PL-based functions and the PS.

There are two types of interfaces between the PL and the PS:

1. Functional interfaces – available for connecting with user-designed IP blocks in the PL
 - a. AXI interconnect
 - b. Extended MIO interfaces for most of the I/O Peripherals
 - c. Interrupts
 - d. DMA flow control
 - e. Clocks
 - f. Debug interfaces
2. Configuration interface – connected to fixed logic within the PL configuration block, providing PS control
 - a. PCAP
 - b. Configuration status
 - c. SEU
 - d. Program/Done/Init

For details on PS-PL interfaces refer to [Chapter 2, Signals, Interfaces, and Pins](#).

Programmable Logic Design Guide

22.1 Introduction

This chapter covers the following topics:

- [Programmable Logic for Software Offload](#) is intended to introduce the user to high-level concepts of using the Programmable Logic (PL) to offload CPU functions.
 - [PL and Memory System Performance Overview](#) discusses various performance-related behaviors of memory paths through the PS.
 - [Choosing a Programmable Logic Interface](#) contrasts different PL interfaces available and shows typical uses.
-

22.2 Programmable Logic for Software Offload

Zynq devices have the unique capability of mapping software algorithms directly to programmable logic. Benefits may include reduced execution time, reduced operating power per function, reduced memory traffic and predictable low latency.

This section describes these benefits from a general perspective. Later sections describe specific performance and potential programmable logic topologies.

22.2.1 Benefits of Using PL to Implement Software Algorithms

Performance

Algorithms implemented in programmable logic can often be scaled to a full parallel implementation delivering maximum throughput, or to an intermediate throughput level at lower area cost. This allows the performance of an algorithm to be scaled well beyond what is achievable on the A9 or NEON units.

For example, consider an algorithm which requires 100 basic operations roughly equivalent to 100 instructions or lines of C code on the A9. A fully parallel programmable logic implementation might implement these operations using LUTs, DSPs and BRAMs. If the PL executes these 100 operations in parallel and is clocked at $\frac{1}{4}$ the rate of the ARM clock, this function would have a potential speedup of 25x. This assumes that the PL implementation is not limited by I/O or resources.

Power

An additional benefit of moving operations to the programmable logic is a reduction in power. Depending on the operations, programmable logic can reduce power per OP by 10-100x. Thus it may be useful to implement algorithms in the PL solely to reduce system power.

One issue to be aware of is that if the algorithm requires access to external memory, the energy cost of accessing the external memory could dominate the energy budget making a reduction in the operation power irrelevant.

Latency

Parallel logic in the PL has a low predictable delay, and cannot be interrupted. For this reason algorithms which are used to respond to real time events originating in the PL might best be serviced by algorithms in the programmable logic. This approach can reduce response time from thousands of clocks to tens of clocks.

22.2.2 Designing PL Accelerators

Programmable logic accelerators are typically created in the RTL languages Verilog or VHDL. Experienced RTL engineers can use C-code as a golden model to create an efficient hardware implementation of the algorithm in programmable logic.

For software programmers who are more comfortable with the C language, C-to-Gates compilers exist which can allow a user to build HW accelerators using the C language. Keeping in mind that C is a sequential language, automated compiler methods can be used to map the sequential code to parallel hardware without user intervention.

For instance, a FOR loop such as `for (i=0; i<10; i++) { x[i]=a[i]+b[i]; }` can be unrolled to create 10 individual adders all operating in parallel.

For video and DSP algorithms, tools such as Matlab Simulink and Xilinx System Generator can be used to directly create logic from algorithmic flowgraphs. A primary advantage of using Matlab Simulink is the rich library of functions which can be used to model, simulate and verify the hardware implementation.

Dataflow

Regardless of how an accelerator or offload engine is designed, once implemented it requires efficient dataflow to and from the accelerator. In many cases, scheduling the dataflow between the accelerator and DRAM can be more of a design challenge than implementing the actual algorithm. The word dataflow is used to reference the motion of data between system memories and PL functional units using AXI interconnect and local interconnect.

22.2.3 PL Acceleration Limits

The achievable speedup of an accelerator can be limited by I/O, resource, and latency requirements.

I/O Rate Limits

A key observation is that processing cannot proceed faster than the speed of the data transfers to and from the functional unit. For functions with a high ratio of operations to I/O the data rates will not be a limiting factor, however for operations with a low ratio of operations to I/O, dataflow will limit the maximum performance attainable.

For example, assume 12 bytes of input data is being read from DDR and 4 bytes of results are written back to DDR. DDR3 at 32-bits, 1,066 Gb/s and 75% utilization is limited to roughly 3.2 GB/s. If 16 bytes are required per operation, the dataflow limits the performance to 3,200/16, or 200M function/sec. Note that this is independent of the complexity of the function. Even a simple 3 input adder is limited by the DDR bandwidth to 200M operations/sec and is not likely to be faster than an ARM A9 CPU. If however, the function consists of several thousands of operations all of which can proceed in parallel, or in a pipelined fashion, then the PL can often achieve speedups of a 10-100x.

Resource Limits

While potential speedup can be quite high, the amount of logic in the PL can limit the achievable speedup. For instance, an application which requires 100 DSPs to achieve a speedup of 24x might be limited to a 12x speedup if only 50 DSPs are available.

Latency Limits

The examples above assume that the PL can effectively proceed without intervention by the ARM processor. This is the case in situations where the PL implements a predetermined algorithm and dataflow using pre-allocated buffers and data is not resident in caches. In cases where the processor is creating data for the PL accelerator, additional CPU tasks might be required before the PL can begin working on the data. The CPU might need to allocate buffers and pass physical buffer addresses to the PL, or data might be flushed from cache to DDR or OCM or signal the PL to start processing. These additional steps add delays (called latency) to the total processing time. If these delays are significant, the potential acceleration is reduced. Typically it takes 100-200 clocks for the ARM processor to write a few words of data to a PL function. In general, CPU to PL calling latency is not a significant impact for applications processing more than 4 KB of data.

22.2.4 Power Offload

The PL can be used to implement individual functions at lower energy cost than when executed on the ARM A9 application processors. Less energy per operation is required because when a function is implemented in the PL, data is transferred from operator to operator in a local assembly line fashion using short, low capacitance local connections.

The same function implemented on a processor requires an instruction and data fetch from local caches or external memory and a result to be written back to registers or the memory system over longer, higher capacitance interfaces. When functions require data to be stored in memory, block

RAM can be used at lower energy cost than processor caches. Table 22-1 summarizes the approximate energy cost for various functions implemented on both the A9 processor and the 7 series programmable logic.

Table 22-1: Estimated Energy Costs for Common Operations

Operation	PL Resource	ARM A9 Resource	ARM A9 energy/OP (pico Joules or mW/GOP/sec)	PL energy/OP (pico Joules op mW/GOP/sec)
Logical Op of 2 var	LUT/FF	ALU		1.3
32-bit ADD	LUT/FF	ALU		1.3
16x16 Mult	DSP	ALU		8.0
32-bit read/write register	LUTRAM	L1		1.4
32-bit read/write AXI register	LUT/FF	AXI		30
32-bit read/write local RAM	BRAM	L2		23.7/17.2
32-bit Read/Write OCM	AXI/OCM	CPU/OCM		44
32-bit Read/Write DDR3	AXI/DDR	CPU/DDR		541/211

Notes:

1. A9 energy costs estimated from ARM power indicative benchmarks.
2. PL Energy costs for custom programmable logic functions are estimated using the Xilinx XPE power estimator http://www.xilinx.com/ise/power_tools/license_7series.htm.

Typically, the energy cost to read or write external DDR memory is roughly the same and much larger than the operation cost. As a consequence the energy required by functions which require even a small percentage of external access is dominated by the energy cost of the external access and the total cost will be the same in both PL and CPU implementations. Thus a key to minimizing energy cost is to localize data movement to the PL. If space allows, rather than storing data structures off chip, store them in OCM, BRAM, LUTRAM or flip-flops and avoid the use of unnecessary storage. This approach may require code to be restructured to avoid the use of unnecessary buffers.

22.2.5 Real Time Offload

The ARM A9 CPUs are optimized for application processing rather than real-time response and are often running an operating system such as Linux. The PL can be used augment the A9s with excellent real time response.

MicroBlaze Assisted Real Time Processing

One or more MicroBlaze processors can be used to as micro controller to manage realtime events. MicroBlaze offers excellent real-time response and can be dedicated to particular tasks. MicroBlaze controllers can typically use a few block RAMs, approximately 2,000 LUTs, and run at 100-200 MHz. Interrupt response times are in the 10s of clocks. Alternately the MicroBlaze can poll for events which can be serviced in just a few clocks. Code can be written in C or for ultimate real time control in assembly. Unlike the Cortex-A9 CPUs, MicroBlaze has a fixed execution pipeline and offers predictable response times. For many applications a PicoBlaze processor may be adequate and uses just a few hundred LUTs.

PL Interrupt Servicing

PS interrupts are routed to the PL and can be serviced by a MicroBlaze processor or by hardware state machines.

HW State Machines

When programmable response times from a MicroBlaze or PicoBlaze CPU are not sufficient, hardware state machines can be created to respond to events. These state machines are generally created in RTL, but can also be generated using MATLAB Simulink and Labview graphical design languages.

22.2.6 Reconfigurable Computing

The programmable logic in each Zynq device can be reconfigured as needed to provide new hardware accelerators. Either the entire device can be reconfigured, or a selected portion of the PL can be reconfigured. This allows for a library of accelerator functions to be stored on disk, flash memory, or DRAM and downloaded on demand. The PS can be used to orchestrate this reconfiguration over the PCAP interface and manage the allocation of PL resources.

Programmable Engines

Typically programmable logic based accelerators implement a specific data flow graph which directly converts input data to output data. An example would be a matrix multiply which pushes data from an input buffer through an array of multipliers and adders and stores the result in a result buffer. An alternative approach might be to build a programmable engine with multipliers and adder instructions to implement the algorithms and general purpose memories to store the data.

While not generally as efficient as fixed function flowgraphs, programmable engines have the advantage of being reprogrammable to implement alternate algorithms. An additional advantage is that they can be used to implement complex functions as the number of operations is limited only by the instruction memory or the bandwidth required to fetch instructions from DRAM. Also a programmable engine may more easily match the required computational rate than a fixed function flowgraph. In general, programmable engines require access to local memories for code and data storage and can require significantly more memory than fixed function flowgraphs as well as additional logic for address generation and instruction decoding.

Open CL has been used as a programming language for these types of engines, but assembly code is also viable. These engines are an area of active research and some IP is now commercially available.

22.3 PL and Memory System Performance Overview

This section provides a comparison of various performance-related behaviors of memory paths through the PS. It is intended to familiarize the designer with the performance-related behaviors of the PL and PS memory system.

22.3.1 Theoretical Bandwidth

Table 22-2 and Table 22-3 provide a basic introduction of relative performance capabilities between various programmable interfaces, DMA, and memory controllers. The bandwidth are calculates as the interface width multiplied by a typical clock rate, not including any protocol overhead. Refer to the appropriate data sheet for actual maximum frequencies.

Table 22-2: Theoretical Bandwidth of PS-PL and PS Memory Interfaces

Interface	Type	Bus Width (bits)	IF Clock (MHz)	Read Bandwidth (MB/s)	Write Bandwidth (MB/s)	R+W Bandwidth (MB/s)	Number of Interfaces	Total Bandwidth (MB/s)
General Purpose AXI	PS Slave	32	150	600	600	1,200	2	2,400
General Purpose AXI	PS Master	32	150	600	600	1,200	2	2,400
High Performance (AFI) AXI_HP	PS Slave	64	150	1,200	1,200	2,400	4	9,600
AXI_ACP	PS Slave	64	150	1,200	1,200	2,400	1	2,400
DDR	External Memory	32	1,066	4,264	4,264	4,264	1	4,264
OCM	Internal Memory	64	222	1,779	1,779	3,557	1	3,557

Table 22-3: Theoretical Bandwidth of PS DMA Controllers

DMA	Type	IF Width (Bits)	IF Clock (MHz)	Read BW (MB/s)	Write BW (MB/s)	R+W BW (MB/s)	Number of Interfaces	Total Bandwidth (MB/s)
DMAC	ARM PL310	64	222	1,776	1,776	3,552	1	3,552
Gigabit Ethernet	PS Master	4	250	125	125	250	2	500
USB	PS Master	8	60	60	60	60	2	120
SD	PS Master	4	50	25	25	25	2	50

Table 22-4: Theoretical Bandwidth of PS Interconnect

Interconnect	Clock Domain	IF Width (Bits)	IF Clock (MHz)	Read BW (MB/S)	Read BW (MB/S)	R+W BW (MB/s)
Central Interconnect	CPU_2x	64	222	1,776	1,776	3,552
Masters	CPU_1x	32	111	444	444	888
Slaves	CPU_1x	32	111	444	444	888
Master Interconnect	CPU_2x	32	222	888	888	1,776
Slave Interconnect	CPU_2x	32	222	888	888	1,776
Memory Interconnect	DDR_2x	64	355	2,840	2,840	5,680

A few performance insights can be inferred from these relative throughputs.

- The OCM or DDR memories are not able to be fully utilized by a single master, except if a low DDR clock rate is used. For example, DDR read bandwidth is limited to 2,840 MB/s on a particular port by the memory interconnect.
- The interconnect generally provides enough bandwidth to sustain access to the memory devices.

22.3.2 DDR Efficiency

One design consideration when using the PL to access external memory is the total amount of DDR memory bandwidth that is available. One useful metric of DDR bandwidth is the efficiency of the controller. Efficiency is the total data passed through the controller versus its theoretical throughput during a test period. Table 22-5 and Table 22-6 lists the efficiency the DDR controller under various access types. The system is configured according to Table 22-7.

Table 22-5: DDR Efficiency (System #1, 4 HP/AFI masters, AXI Burst Length of 16)

Access Type	Address Pattern	Efficiency (%)
Reads	Sequential	97
Reads	Random	92
Writes	Sequential	90
Writes	Random	87
Reads and Writes	Sequential	87
Reads and Writes	Random	79

From a design planning perspective, accesses tested in Table 22-5 could be described as optimistic to near-typical values. The random read/write pattern is not worst case as the DDR controller optimization features are still able to improve efficiency; there may be other more pessimistic access patterns. Overall, the DDR controller was designed to have a maximum efficiency of approximately 75%.

Table 22-6 lists a DDR efficiency versus burst length example. It illustrates that moderate length bursts do not result in significant DDR efficiency loss. These moderate burst lengths can be useful in latency-sensitive environments where longer bursts can increase latency for higher-priority masters in the system.

Table 22-6: DDR Efficiency versus AXI Burst Length (System #1, 4 HP/AFI masters, Sequential Read/Writes)

Burst Length	DDR Efficiency (%)
4	87
8	87
16	87

Table 22-7: Latency Example Measurement Systems

System	PL AXI Clock (MHz)	CPU_6x4x (MHz)	CPU_2x (MHz)	DDR_3x (MHz)	DDR_2x (MHz)	DRAM	DRAM (Mb/s)
#1	150	675	225	525	350	DDR3	1,050

22.3.3 OCM Efficiency

A similar efficiency test to the DDR example above using four high-performance ports to OCM show a maximum efficiency of 80%.

22.3.4 Interconnect Throughput Bottlenecks

At typical high-performance clock ratios, the PS interconnect is not typically the limiting factor in a high-performance system. One exception to this is when using two high-performance (HP/AFI) ports to DDR. Since ports 0/1 and 2/3 are arbitrated before the DDR controller, it is beneficial in the two port case to use one port each from these pairs, such as port 0 and 2.

22.4 Choosing a Programmable Logic Interface

This section discusses various options to connecting Programmable Logic (PL) to the Processing System (PS). The main emphasis is on data movement tasks such as direct memory access (DMA).

22.4.1 PL Interface Comparison Summary

Table 22-8 presents a qualitative overview of data transfer use cases. The estimated throughput column reflects suggested maximum throughput in a single direction (read/write).

Table 22-8: Data Movement Method Comparison Summary

Method	Benefits	Drawbacks	Suggested Uses	Estimated Throughput
CPU Programmed I/O	<ul style="list-style-type: none"> Simple Software Least PL Resources Simple PL Slaves 	<ul style="list-style-type: none"> Lowest Throughput 	<ul style="list-style-type: none"> Control Functions 	<25 MB/s
PS DMAC	<ul style="list-style-type: none"> Least PL Resources Medium Throughput Multiple Channels Simple PL Slaves 	<ul style="list-style-type: none"> Somewhat complex DMA programming 	<ul style="list-style-type: none"> Limited PL Resource DMAs 	600 MB/s
PL AXI_HP DMA	<ul style="list-style-type: none"> Highest Throughput Multiple Interfaces Command/Data FIFOs 	<ul style="list-style-type: none"> OCM/DDR access only More complex PL Master design 	<ul style="list-style-type: none"> High Performance DMA for large datasets 	1,200 MB/s (per interface)
PL AXI_ACP DMA	<ul style="list-style-type: none"> Highest Throughput Lowest Latency Optional Cache Coherency 	<ul style="list-style-type: none"> Large burst may cause cache thrashing Shares CPU Interconnect bandwidth More complex PL Master design 	<ul style="list-style-type: none"> High Performance DMA for smaller, coherent datasets Medium granularity CPU offload 	1,200 MB/s
PL AXI_GP DMA	<ul style="list-style-type: none"> Medium Throughput 	<ul style="list-style-type: none"> More complex PL Master design 	<ul style="list-style-type: none"> PL to PS Control Functions PS I/O Peripheral Access 	600 MB/s

22.4.2 Cortex-A9 CPU via General Purpose Masters

The least intrusive method from a software perspective is to use the Cortex-A9 to move data between the PS and PL (see Figure 22-1). Data flow is directly moved by a CPU, removing the need to handle events from a separate DMA. Access to the PL is provided through the two M_AXI_GP master ports, which each have a memory address range to originate PL AXI transactions. The PL design is also simplified since as little as a single AXI slave can be implemented to service the CPU requests.

Drawbacks of using a CPU to move data is that a sophisticated CPU is spending cycles performing simple data movement instead of complex control and computation tasks, and the limited throughput available. Transfer rates less than 25 MB/s are reasonable with this method.

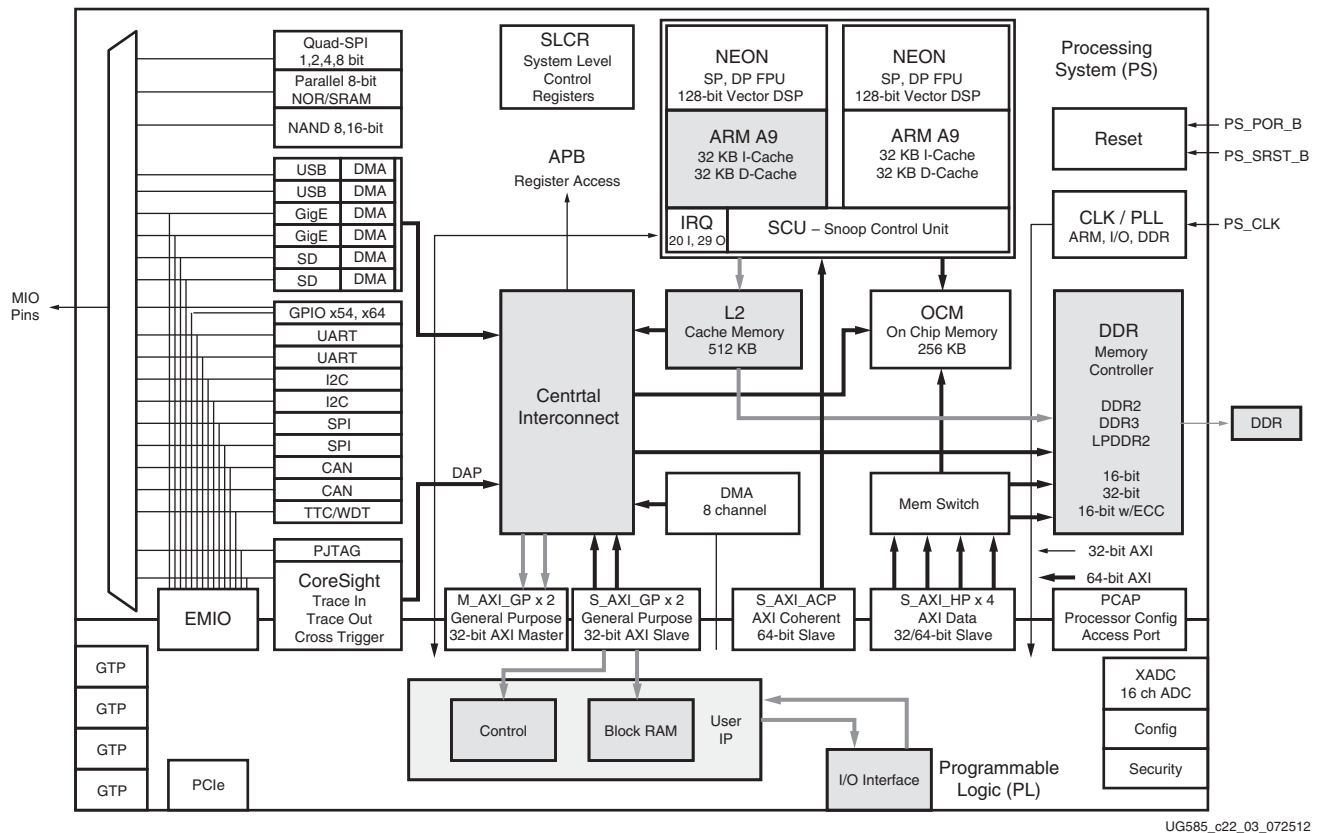


Figure 22-1: Example Cortex-A9 PL Data Movement Topology

22.4.3 PS DMA Controller (DMAC) via General Purpose Masters

The PS DMA controller (DMAC) provides a flexible DMA engine that can provide moderate levels of throughput with little PL logic resource usage (see Figure 22-2). The DMAC resides in the PS and must be programmed via DMA instructions residing in memory, typically prepared by a CPU. With support for up to eight channels, multiple DMA fabric cores can potentially be served in the single DMAC. However, the flexible programmable model may increase software complexity relative to CPU transfer or specialized PL DMA.

The DMAC interface to the PL is through the general purpose AXI master interfaces, whose 32-bit width along with the centralized DMA nature (a read and write transaction for each movement) of the DMAC to limit the DMAC from highest throughput. A peripheral request interface also allows PL slaves to provide status to the DMAC on buffer state, to prevent transactions involving a stalled PL peripheral from unnecessarily also stalling interconnect and DMAC bandwidth.

See Chapter 9, DMA Controller for more information on the DMAC controller. More information on the M_AXI_GP interfaces can be found in Chapter 5, Interconnect.

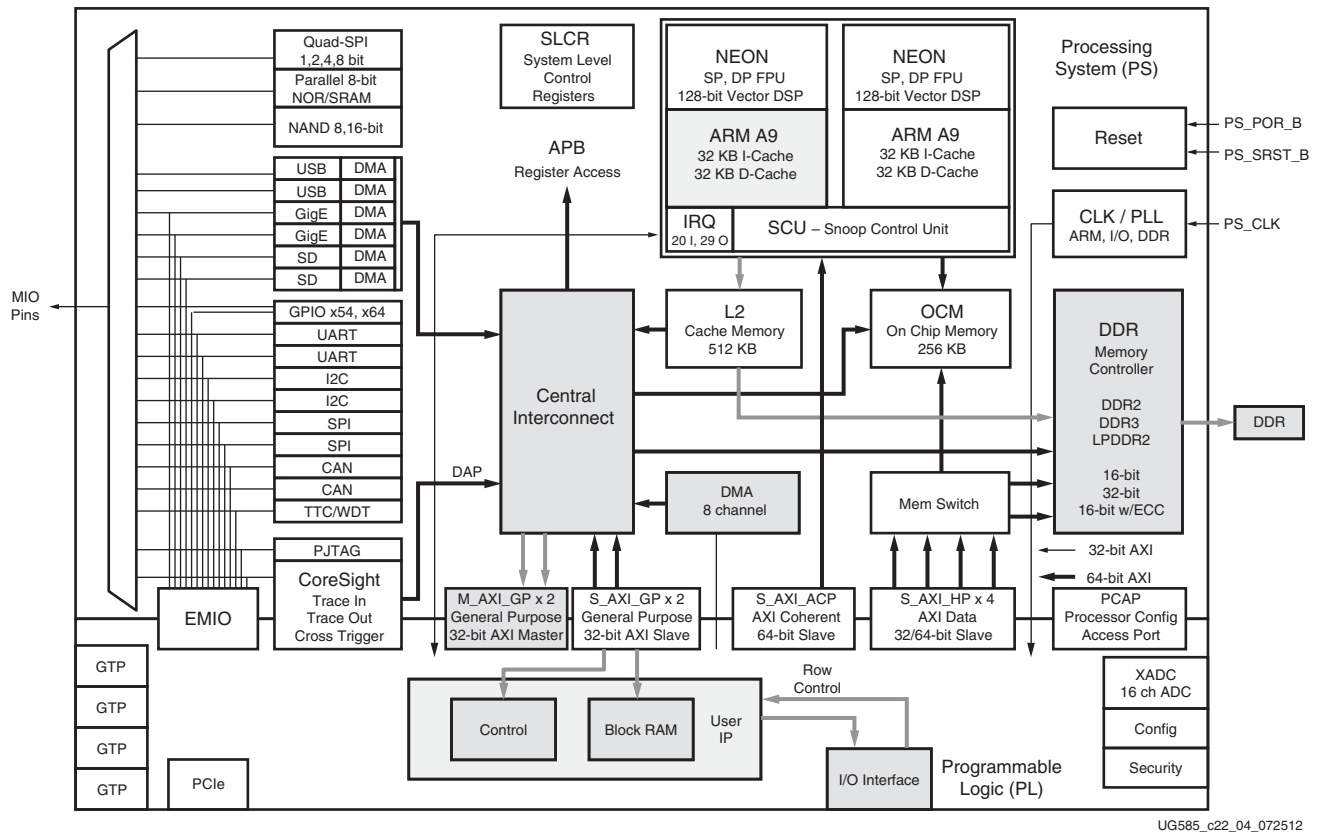


Figure 22-2: Example DMAC DMA Topology

22.4.4 PL DMA via AXI High-Performance (HP) Interface

The high-performance (S_AXI_HP) PL interfaces provide high-bandwidth PL slave interfaces to OCM and DDR memories. The AXI_HP ports are unable to access any other slaves. With four, 64-bit wide interfaces, the AXI_HP provide the greatest aggregate interface bandwidth. The multiple interfaces also save PL resources by reducing the need to a PL AXI interconnect. Each AXI_HP contains control and data FIFOs to provide buffering of transactions for larger sets of bursts, making it ideal for workloads such as video frame buffering in DDR. This additional logic and arbitration does result in higher minimum latency than other interfaces.

The user IP logic residing in the PL will generally consist of a low-speed control interface and higher performance burst interface, as shown in Figure 22-3. If control flow is orchestrated by the Cortex-A9 CPU, the general purpose M_AXI_GP port can be used for tasks such as configuring the memory addresses the User IP should access and transaction status. Transaction status can also be conveyed via PL to PS interrupts. Higher performance devices connected to AXI_HP should be able to issue multiple outstanding transactions to take advantage of the AXI_HP FIFOs.

The PL design complexity of multiple AXI interfaces along with the associated PL utilization are the primary drawbacks of implementing a DMA engine in the PL for both S_AXI_HP and S_AXI_ACP interfaces.

See Chapter 5, *Interconnect* for more information on the AXI_HP interface.

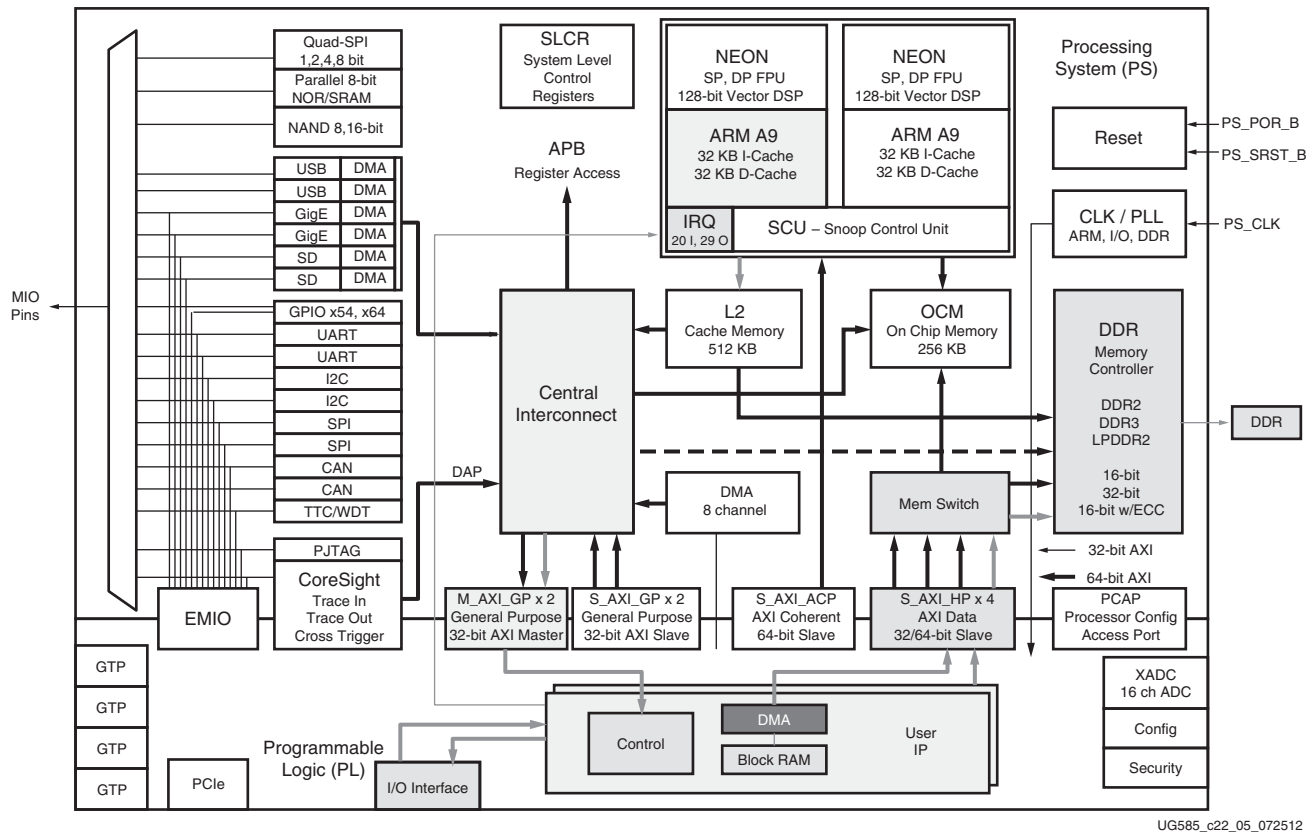


Figure 22-3: Example High-Performance (HP) DMA Topology

22.4.5 PL DMA via AXI ACP

The AXI ACP interface (S_AXI_ACP) provides a similar user IP topology as the high performance S_AXI_HP interfaces. Also 64 bits wide, the ACP also provides highest throughput capability for a single AXI interface. As shown in Figure 22-4, the User IP topology is likely often similar to the S_AXI_HP example in the previous section.

The ACP differs from the HP performance ports due to its connectivity inside of the PS. The ACP connects to the snoop control unit (SCU) which is also connected to the CPU L1 and the L2 cache. This connectivity allows ACP transactions to interact with the cache subsystems, potentially decreasing total latency for data to be consumed by a CPU. These optionally cache-coherent operations can prevent the need to invalidate and flush cache lines. The ACP also has the lowest memory latency to memory of the PL interfaces. The connectivity of the ACP is similar to that of the CPUs.

The drawbacks from using the ACP besides those shared with the S_AXI_HP interfaces also stem from the locality to the cache and CPUs. Memory accesses through the ACP utilize the same interconnect paths as the APU, potentially decreasing CPU performance. Large, coherent ACP transfers can cause thrashing of the cache. Thus ACP coherent transfers are best suited for less than the largest data-sets. The ACP low-latency access allows opportunity for algorithm acceleration of medium granularity.

For more information on S_AXI_ACP, see [Chapter 3, Application Processing Unit](#). See [Chapter 29, On-Chip Memory \(OCM\)](#) when using ACP with OCM.

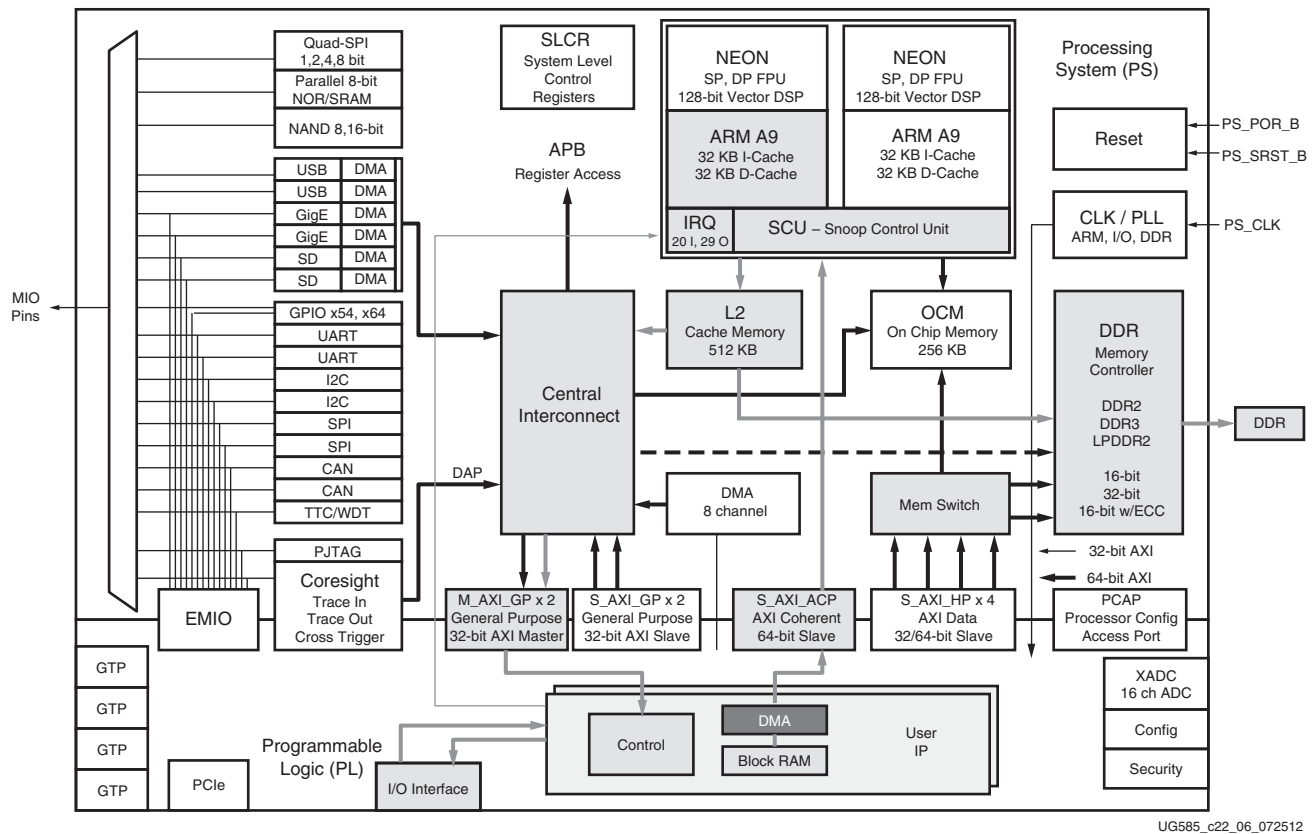


Figure 22-4: Example ACP DMA Topology

22.4.6 PL DMA via General Purpose AXI Slave (GP)

While the general purpose AXI Slave (S_AXI_GP) has reasonably low latency to OCM and DDR, its narrow 32-bit interface limits its utility as a DMA interface. The two S_AXI_GP interfaces are more likely to be used for lower-performance control access to the PS memories, registers and PS peripherals.

More information on the S_AXI_GP interfaces can be found in [Chapter 5, Interconnect](#).

Programmable Logic (PL) Test and Debug

23.1 Introduction

Zynq-7000 EPP devices provides extensive debug capability for accessing the PS debug structure (see [Chapter 28, System Test and Debug](#)) from the PL. This allows for integrated test and debug on both PS and PL simultaneously.

Xilinx provides the fabric trace monitor (FTM) for programmable logic test and debug. It is based on the ARM CoreSight architecture, and is a component of the trace source class (see [Chapter 28, System Test and Debug](#)) in the CoreSight system within Zynq-700 EPP devices. The FTM receives trace data from the PL and formats it into trace packets to be combined with the trace packets from other trace source components such as PTM and ITM. With this capability, PL events can easily be traced simultaneously with PS events. The FTM also supports cross-triggering between the PS and PL, except for the trace dumping feature. In addition, the FTM provides general-purpose debug signals between the PS and PL.

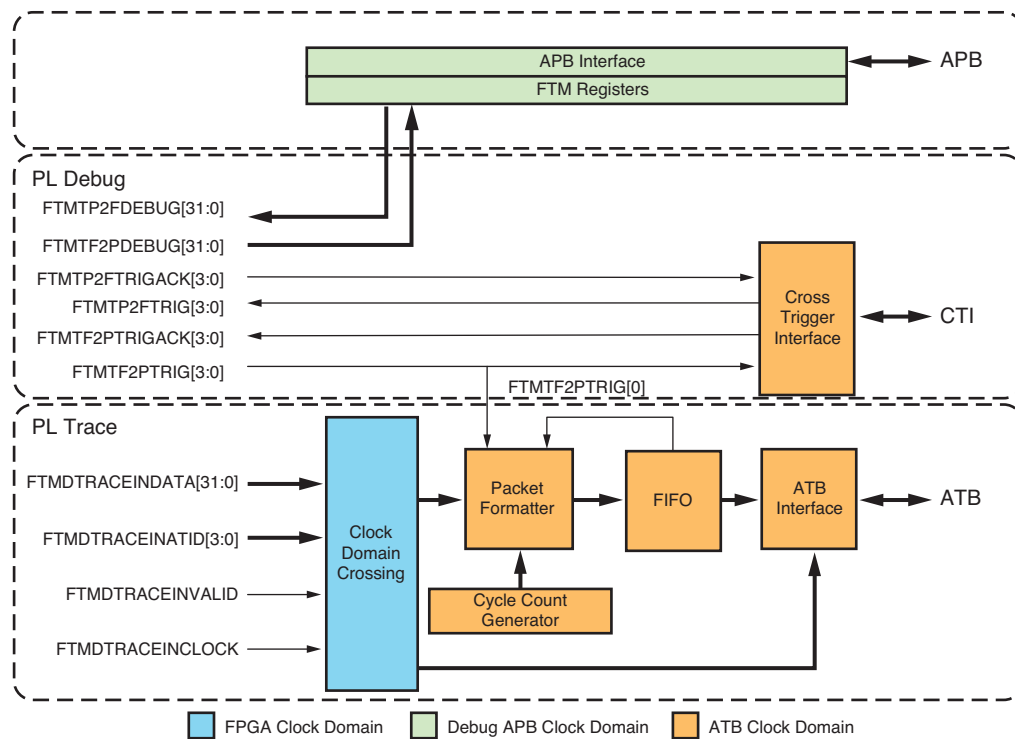
23.1.1 Features

The key features of the PL test and debug are as follows:

- ARM CoreSight compatible
- 32-bit trace data from the PL
- 4-bit trace ID from the PL
- Clock domain crossing between the PL and PS
- FIFO buffering for trace packets to absorb bursts of trace data from the PL
- Indication of FIFO overflow via generation of an overflow packet
- Trace packets are compatible with ARM trace port software and hardware
- Trigger signals to/from the PL
- General-purpose I/Os to/from the PL

23.1.2 Block Diagram

A block diagram of the FTM is shown in Figure 23-1.



UG585_c23_01_030312

Figure 23-1: FTM Block Diagram

As shown in Figure 23-1, the following functional blocks comprise the FTM:

- APB Interface
 - This is the interface to the CoreSight debug APB, through which CPUs and JTAG can interact with the FTM.
- FTM Registers
 - These are programmable registers.
- Clock Domain Crossing
 - This block synchronizes signals between the PL clock domain and the PS clock domain.
- PL Debug Ports
 - This block provides:
 - General purpose I/Os, 32 bits to the PL and 32 bits from the PL. These are accessed through reads and writes to registers.
 - Trigger signals, 4 pairs to the PL and 4 pairs from the PL. Each pair consists of a trigger signal and an acknowledge signal, and follows ARM standard CTI handshake protocol.

- Trigger input 0 (FTMTF2PTRIG[0]) can be used to generate trigger packets.
- Packet Formatter
 - This block is responsible for gathering trace data and formatting the data into trace packets. In addition to trace packets, the packet formatter can also generate various other types of packets to convey additional information to the CoreSight system within the PS.
- Cycle Count Generator
 - This block is used to provide a binary count value for time stamping packets. This is achieved by a counter, which is:
 - 32-bit, free-running, clocked by CPU_2x
 - Pre-scaled by $2^{\text{CYCOUNTPRE}}$ (range:1 to 32,768)
 - Reset by POR, FTMGLBCTRL[FTMENABLE]=0, CoreSight reset request through JTAG
- FIFO
 - The FIFO is used to buffer packets before they are sent to the ATB. The FIFO has these properties:
 - 64-packets deep
 - When the FIFO overflows, it signals the packet formatter to generate an overflow packet. In this case, some trace data is lost.
- ATB Interface
 - This is the interface to the CoreSight ATB, over which packets are sent.
- Cross Trigger Interface
 - This block is the interface to the CoreSight ECT system (see [Chapter 28, System Test and Debug](#)).

23.1.3 System Viewpoint

For details on how the FTM is connected to, and interacts with, the rest of the CoreSight system within the PS, see [Chapter 28, System Test and Debug](#).

23.2 Functional Description

23.2.1 Basic Operation

The PL trace module captures the trace data from the PL. The user supplies the trace data, trace ID, valid, and clock signals to the FTM at the PL-PS boundary. All data, ID, and valid signals must be stable on the rising edge of the FTMDTRACEINCLOCK signal for the FTM to correctly sample them.

When FTMDTRACEINVALID is asserted, the PL trace signals are available to the clock domain crossing interface, which synchronizes the data and ID, and sends them to the packet formatter.

The packet formatter generates packets and submits them to the FIFO. The packet formatter can generate the following types of packets:

- Trace packets: These packets are generated when valid trace data is available from the PL.
- Trigger packets: These trigger packets can only be generated by the FTMTF2PTRIG[0] signal.
- Cycle count packets: These packets provide continuous timestamps that are used to reconstruct a real-time trace.
- Overflow packets: These packets are generated when the FIFO overflows.
- Synchronization packets: These packets are for packet analysis tools to re-align to packet boundaries.

The cross trigger interface communicates with the CoreSight ECT structure. This interface passes re-synchronized trigger signals between the PL and the ECT. This provides the capability of cross-triggering each other between the PS and PL.

23.2.2 Packet Generation

The following are some common scenarios that illustrate packet generation by the FTM.

Typical Case – Trace Enabled, Cycle Count Enabled

When a valid PL trace (FTMDTRACEINVALID is asserted) is captured, a trace packet and a cycle count packet are generated and sent to the ATB. When a trigger occurs (via the FTMTF2PTRIG[0] signal), a trigger packet and a cycle count packet are generated and sent to the ATB.

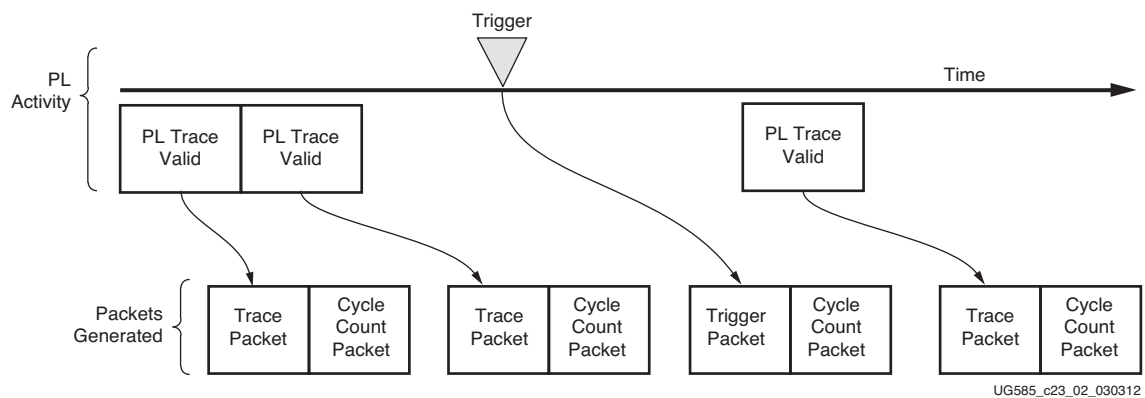


Figure 23-2: Packet Generation, Typical Case

No Cycle Count Case – Trace Enabled, Cycle Count Disabled

This is similar to the preceding case, except that cycle count packets are not generated.

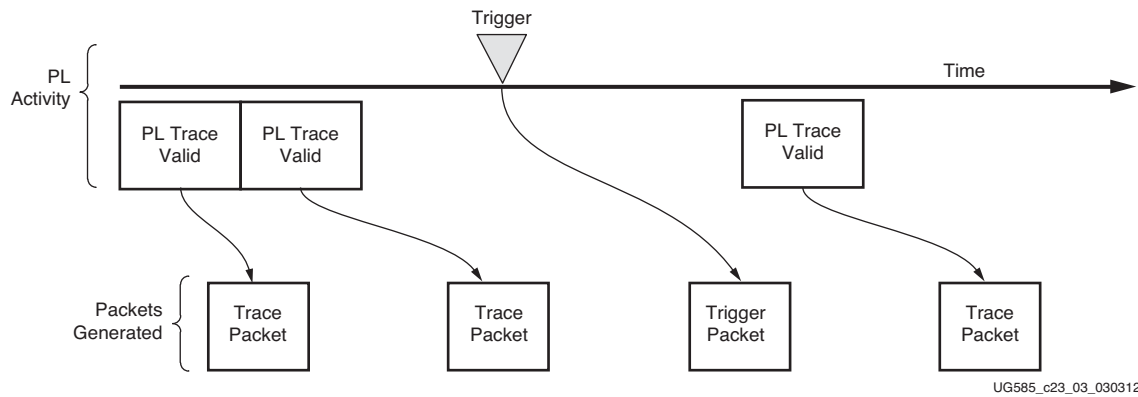


Figure 23-3: Packet Generation, No Cycle Count Case

FIFO Overflow Case – Lost Trace

In this scenario, trace packets are not generated until the FIFO is no longer in an overflow condition. An overflow packet is generated to indicate that there was an overflow condition and some trace packets are lost.

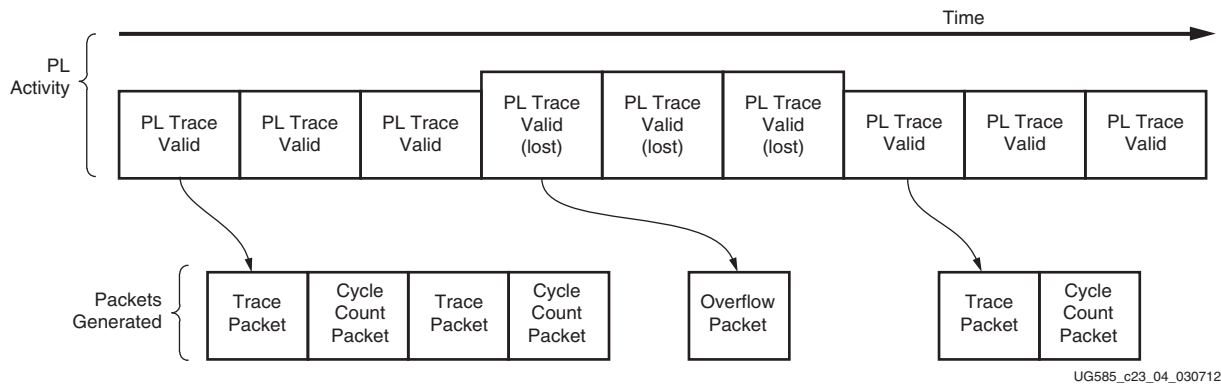


Figure 23-4: Packet Generation, FIFO Overflow Case

Synchronization Case

This scenario illustrates how a synchronization packet is generated amid other types of packets. The FTMSYNCRELOAD register sets the number of packets for which a synchronization packet must be generated.

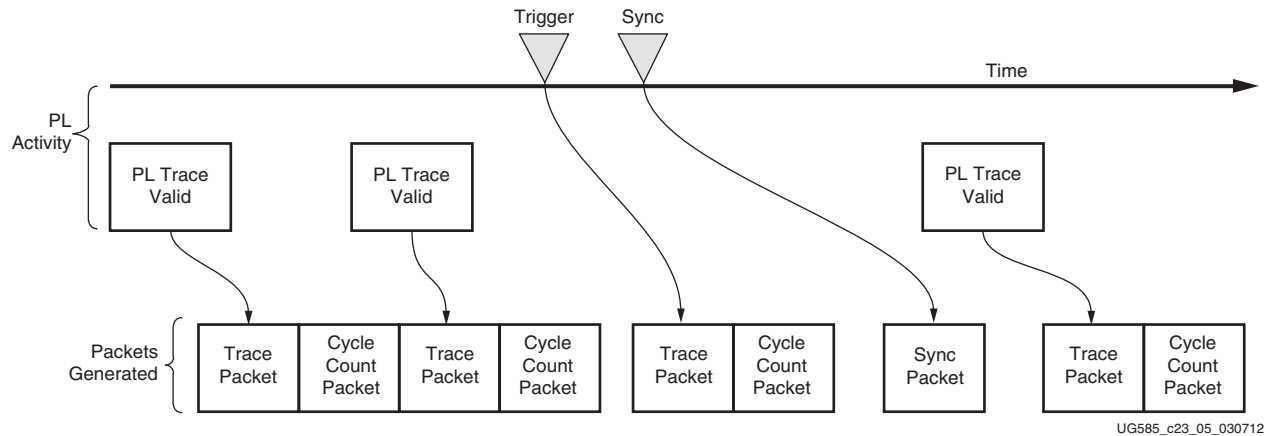


Figure 23-5: Packet Generation, Synchronization Case

23.2.3 Packet Format

General Format

A packet consists of a number of bytes. Table 23-1 shows the basic format of a packet. Except for the synchronization packet and the cycle count packet, the first and the last bytes of a packet have their MSB set to 0 to signify the start/stop of the packet; all bytes between the first and the last bytes have their MSB set to 1.

Table 23-1: General Packet Format

Byte	[7]	[6:0]
0	0	type
1	1	data
...	1	data
n	0	data

Table 23-2 shows the encoding values for the “type” byte.

Table 23-2: “Type” Byte Encoding

Type	[7]	[6:3]	[2:0]
Trace packet	0	trace[3:0]	101
Trigger packet	0	0100	000
Cycle count packer	1	count[3:0]	100

Table 23-2: “Type” Byte Encoding (Cont’d)

Type	[7]	[6:3]	[2:0]
FIFO overflow packet	0	1101	000
Synchronization packet	0	0000	000

Trace Packet

A trace packet contains the 32-bit value from each captured FTMDTRACEINDATA[31:0] from the PL. The MSB of the last byte is determined by the presence of an immediately following cycle count packet. If a cycle count packet follows, the MSB is 1, otherwise it is 0.

Table 23-3: Trace Packet Format

Byte	[7]	[6:0]
0	0	data[3:0], 101
1	1	data[10:4]
2	1	data[17:11]
3	1	data[24:18]
4	count	data[31:25]

Trigger Packet

A trigger packet is generated for each acknowledged trigger input [0] from the PL, i.e., when both FTMTF2PTRIG[0] and FTMTF2PTRIGACK[0] are High.

Table 23-4: Trigger Packet Format

Byte	[7:0]
0	0x20
1	0xA0
2	0xA0
3	0x20 or 0xA0

Cycle Count Packet

A cycle count packet is generated when FTMCONTROL[CYCEN] is set, and a trace packet or a trigger packet is generated. It contains a binary count value taken from the current value of the internal 32-bit free-running counter. It is always a continuation of an immediately preceding trace packet or trigger packet.

Table 23-5: Cycle Count Packet Format

Byte	[7]	[6:0]
0	1	count[3:0], 100
1	1	count[10:4]
2	1	count[17:11]

Table 23-5: Cycle Count Packet Format (Cont'd)

Byte	[7]	[6:0]
3	1	count[24:18]
4	0	count[31:25]

FIFO Overflow Packet

A FIFO overflow packet is generated when a FIFO overflow occurs.

Table 23-6: Overflow Packet Format

Byte	[7:0]
0	0x68
1	0xE8
2	0xE8
3	0x68

Synchronization Packet

A synchronization packet allows the packet analysis tools to periodically re-align to correct packet boundaries, because the packet formatter does not maintain 32-bit word boundaries for packets. The synchronization packet follows the same format as other CoreSight components.

Table 23-7: Synchronization Packet Format

Byte	[7:0]
0-7	0x00
8	0x80

23.3 Signals

The FTM control signals are described in the following sections.

23.3.1 General-Purpose Debug Signals

Table 23-8: General-Purpose Debug Signals

Group	PS-PL Signal	IO	Description
General purpose debug output	FTMP2FDEBUG[7:0]	O	The FTMP2FDBG0 register controls its value.
	FTMP2FDEBUG[15:8]	O	The FTMP2FDBG1 register controls its value.
	FTMP2FDEBUG[23:16]	O	The FTMP2FDBG2 register controls its value.
	FTMP2FDEBUG[31:24]	O	The FTMP2FDBG3 register controls its value.

Table 23-8: General-Purpose Debug Signals

Group	PS-PL Signal	IO	Description
General purpose debug input	FTMTF2PDEBUG[7:0]	I	The FTMTF2PDBG0 register shows its value.
	FTMTF2PDEBUG[15:8]	I	The FTMTF2PDBG1 register shows its value.
	FTMTF2PDEBUG[23:16]	I	The FTMTF2PDBG2 register shows its value.
	FTMTF2PDEBUG[31:24]	I	The FTMTF2PDBG3 register shows its value.

23.3.2 Trigger Signals

Table 23-9: Trigger Signals

Group	PS-PL Signal	IO	Description
Trigger from PS to PL	FTMTP2FTRIG[3:0]	O	Each bit is an asynchronous trigger signal from the CoreSight ECT structure in the PS to the PL. Users must program the CTI connected to the FTM to enable these trigger output signals.
	FTMTP2FTRIGACK[3:0]	I	Each bit is the asynchronous acknowledge signal for the corresponding FTMTP2FTRIG signal.
Trigger from PL to PS	FTMTF2PTRIG[3:0]	I	Each bit is an asynchronous trigger signal from the PL to the CoreSight ECT structure in the PS. Users must program the CTI connected to FTM to enable these trigger input signals.
	FTMTF2PTRIGACK[3:0]	O	Each bit is the asynchronous acknowledge signal for the corresponding FTMTF2PTRIG signal.

23.3.3 Trace Signals

Table 23-10: Trace Signals

Group	PS-PL Signal	IO	Description
Trace from PL to PS	FTMDTRACEINCLOCK	I	Clock signal for the trace data interface. Asynchronous to the PS.
	FTMDTRACEINVALID	I	When this signal is sampled High by the PS using FTMDTRACEINCLOCK, the values on TRMDTRACEINDATA and FTMDTRACEINATID are valid.
	FTMDTRACEINDATA[31:0]	I	Trace data. All 32 bits must be provided.
	FTMDTRACEINATID[3:0]	I	Trace ID to be carried over to the ATB. All 4 bits must be provided.

23.4 Register Overview

Table 23-11: Register Overview

Function	Name	Overview
Control	FTMGLBCTRL FTMCONTROL	Enable FTM, enable cycle count packets, enable trace packets.
Status	FTMSTATUS	Idle status, security signal values, FIFO full/empty.
General debug	FTMP2FDBG FTMF2PDGB	Set the values of the signals presented to the PL. Read the values of the signals from the PL.
Cycle counter prescaler	FTMCYCCOUNTPRE	Set the prescaler value for the cycle counter.
Synchronization counter	FTMSYNCRELOAD FTMSYNCCOUNT	Set how often synchronization packets should be generated.
Configuration	FTMATID	Set the ATID value to the ATB bus.
CoreSight management	Peripheral ID Component ID Device ID, type Authentication Integration test	These registers provide: <ul style="list-style-type: none"> • Identification information • Authentication and access control • Integration test

23.5 Programming Model

23.5.1 FTM Security

FTM security is controlled through the use of the standard CoreSight mechanisms, the SPNIDEN, SPIDEN, NIDEN, and DBGEN signals from the SLCR.

The following actions are taken by the FTM when the corresponding signals are asserted:

- SPNIDEN: FTM trace operations can be enabled.
- SPIDEN: The FTMP2FDBG registers can be modified.
- NIDEN: No effect.
- DBGEN: No effect.

Power Management

24.1 Introduction

This chapter describes the voltage and power management aspects of the Zynq-7000 EPP devices.

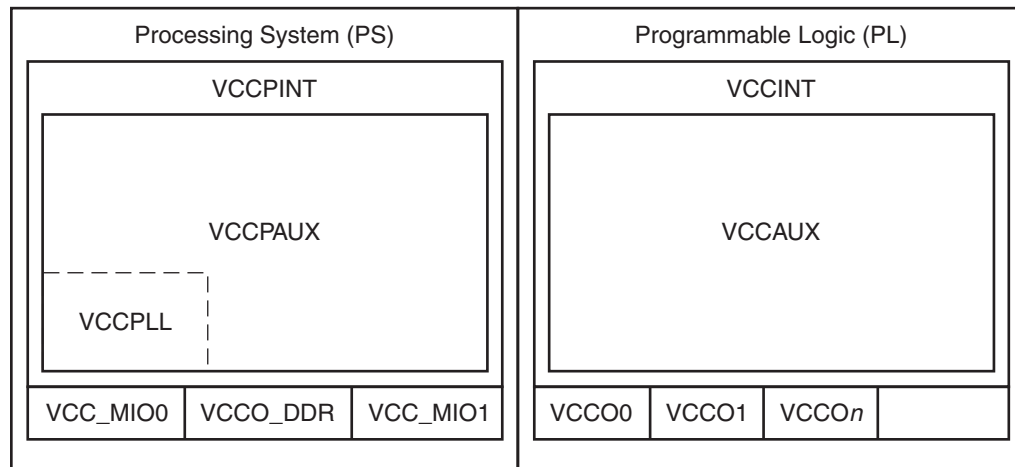
Power is divided into two primary power domains. One domain is reserved for the Processing System (PS) while the other is reserved for the Programmable Logic (PL). Additional power granularity is available via the SelectIO power rails within the PL and MIO power rails in the MIO banks.

The Zynq-7000 EPP split power domain scheme provides designers with the flexibility to save power by utilizing the programmable logic only when needed by their system. Zynq-7000 EPP devices support low-power sleep mode and dynamic power reduction (inside the processing system) to save additional power.

Zynq-7000 EPP devices support power management monitoring by utilizing the sensors in the Xilinx ADC (XADC) block. This block allows for real-time monitoring of voltage and temperature levels within the device.

24.2 Voltage Domains

Figure 24-1 shows the PS and PL separate voltage domains for managing power.



UG585_c24_01_022912

Figure 24-1: Voltage Domains

24.3 Features

Key features of the Zynq-7000 EPP power system include:

- PL power-off support
- Low-power sleep mode support
- PS dynamic power reduction support
- Power monitoring support via XADC
- Independent MIO DDR memory bank
- PS MIO support for HSTL_18, LVTTTL, LVCMOS_18, LVCMOS_25, and LVCMOS_33
- PL SelectIO structure based on standard 7-series architecture

24.4 PS Dynamic Power Reduction in Running Mode

The PS supports many clock domains, each with independent clock gating control. When the system is in running mode, it allows the user to shut down the clock domains that are not used to reduce dynamic power dissipation. Separate clock domains with clock gating control for the PS allow the user to shut down clock domains independently. In normal running mode, the bus clock can be shut

down automatically when the CPU enters WFI or WFE modes. The PS software must ensure that no other bus masters within the PS are using the bus before this mode can be enabled.

24.5 Sleep Mode Sequence

In sleep mode, most functional clock groups are turned off or powered off. The only required active devices are one CPU, the snoop control unit (SCU), and a wakeup device. The wakeup device can be CAN, Ethernet, GPIO, or any device that can generate an interrupt.

Ideally, dynamic power should only use a small portion of the CPU's consumption to monitor wakeup interrupts, the SCU, and the wakeup peripheral device. If both CPUs are running, the master CPU should shut down the secondary CPU before proceeding with the steps described below. The user can select which of the two CPUs to be enabled as the master CPU. A CPU must execute the following steps to enter sleep mode from normal running mode:

1. Wait for all outstanding requests, especially AXI interconnect traffic, to be cleared.
2. Load the wakeup ISR into internal OCM RAM for future wakeup processes.
3. Put the external DDR memory into self-refresh mode.
4. Turn off the DDR, secondary CPU clock, and non-wake up peripheral clocks.
5. Configure I/O properly to minimize power dissipation (e.g., disable DDR termination).
6. Enable the WFI bus clock disable control register bit in the SLCR.
7. Put the PLL into bypass mode.
8. Shut down the PLL.
9. Increase the clock divider to slow down the CPU clock.
10. Execute the WFI instruction to enter WFI mode (the bus clock shuts down a few cycles later).
11. To exit from sleep mode:
 - a. The wakeup device sends an interrupt to the interrupt controller within the SCU.
 - b. The SCU sends an interrupt to the CPU to wake up the primary CPU.
 - c. CPU wake up and bus clock are restored automatically.
12. At this point, the CPU:
 - a. Restores clock dividing settings to normal.
 - b. Powers on the PLLs and bandgap circuit.
 - c. Waits for PLL power-on and lock.
 - d. Disables PLL bypass mode.
 - e. Restores I/O states.
 - f. Enables all required peripheral devices, including DDR controller clocks.
 - g. Sends a command to the external DDR controller to exit self-refresh mode.

Clocks

25.1 Introduction

All of the clocks generated by the PS clock subsystem are derived from one of three programmable PLLs: CPU, DDR and I/O. Each of these PLLs is loosely associated with the clocks in the CPU, DDR and peripheral subsystems.

25.1.1 Block Diagram

The major components of the clock subsystem are shown in Figure 25-1.

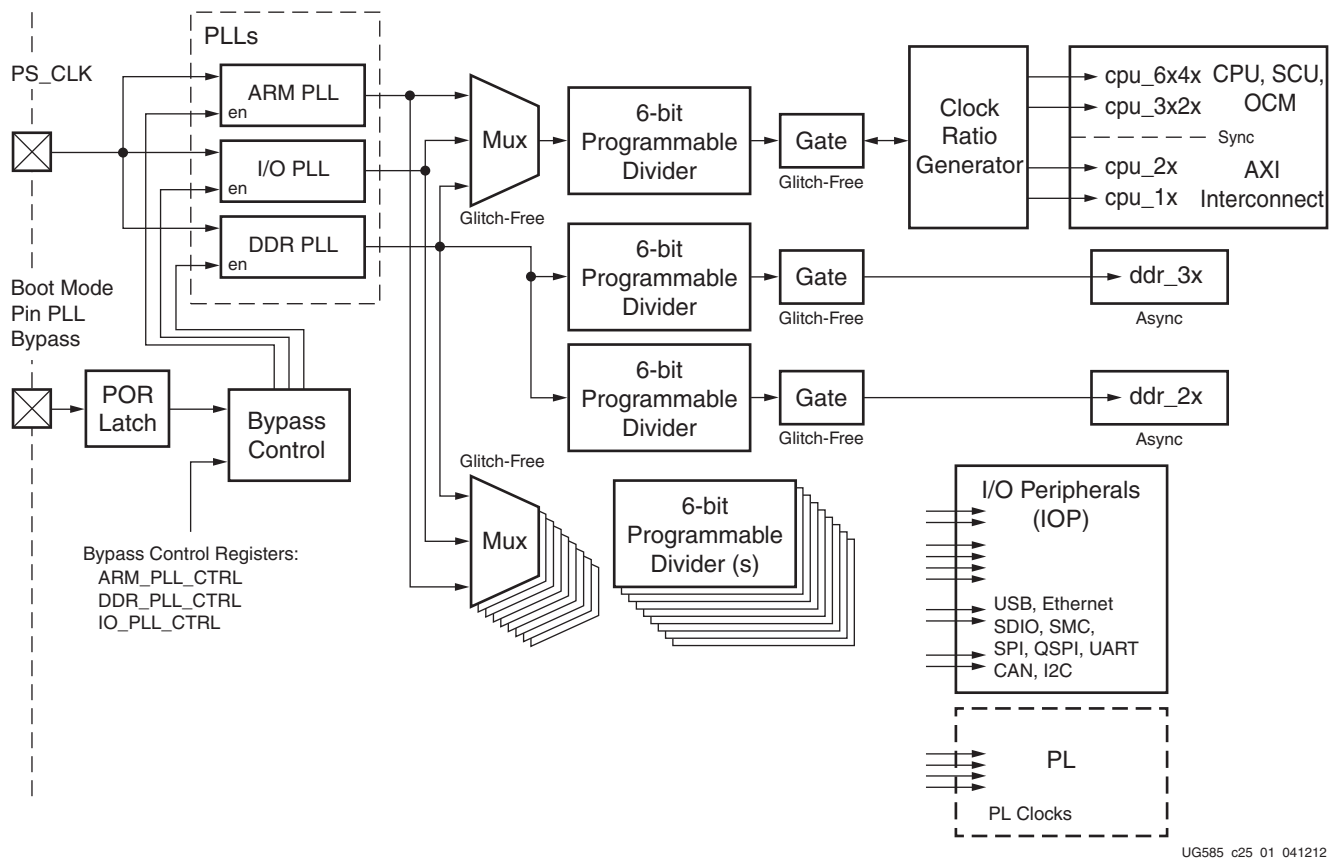


Figure 25-1: Clock Subsystem and Distribution Block Diagram

25.1.2 Clock Generation

During normal operation, the PLLs are enabled, driven by the PS_CLK clock pin. In bypass mode, the clock signal on the PS_CLK pin provides the source for the various clock generators instead of the PLLs. (Refer to the applicable Zynq-7000 EPP data sheet for PS_CLK characteristics.)

When the PS_POR reset signal deasserts, the PLL bypass boot mode pin is sampled and selects between PLL bypass and PLL enabled for all three PLLs. The bypass mode runs the system significantly slower than normal mode, but is useful for low-power applications and debug. After the boot process and when the user code executes, the bypass mode and output frequency of each PLL can be individually controlled by software.

The clock generation paths include glitch-free multiplexers and glitch-free clock gates to support dynamic clock control.

Three Programmable PLLs

- Single external reference clock input for all three PLLs
 - ARM PLL: Usual clock source for the CPUs and the interconnect
 - DDR PLL: Usual clock for the DDR DRAM controller and AXI_HP interfaces
 - I/O PLL: Usual clock for I/O peripherals
- Individual PLL bypass control and frequency programming
- Shared bandgap reference voltage circuit for VCOs

Clock Branches

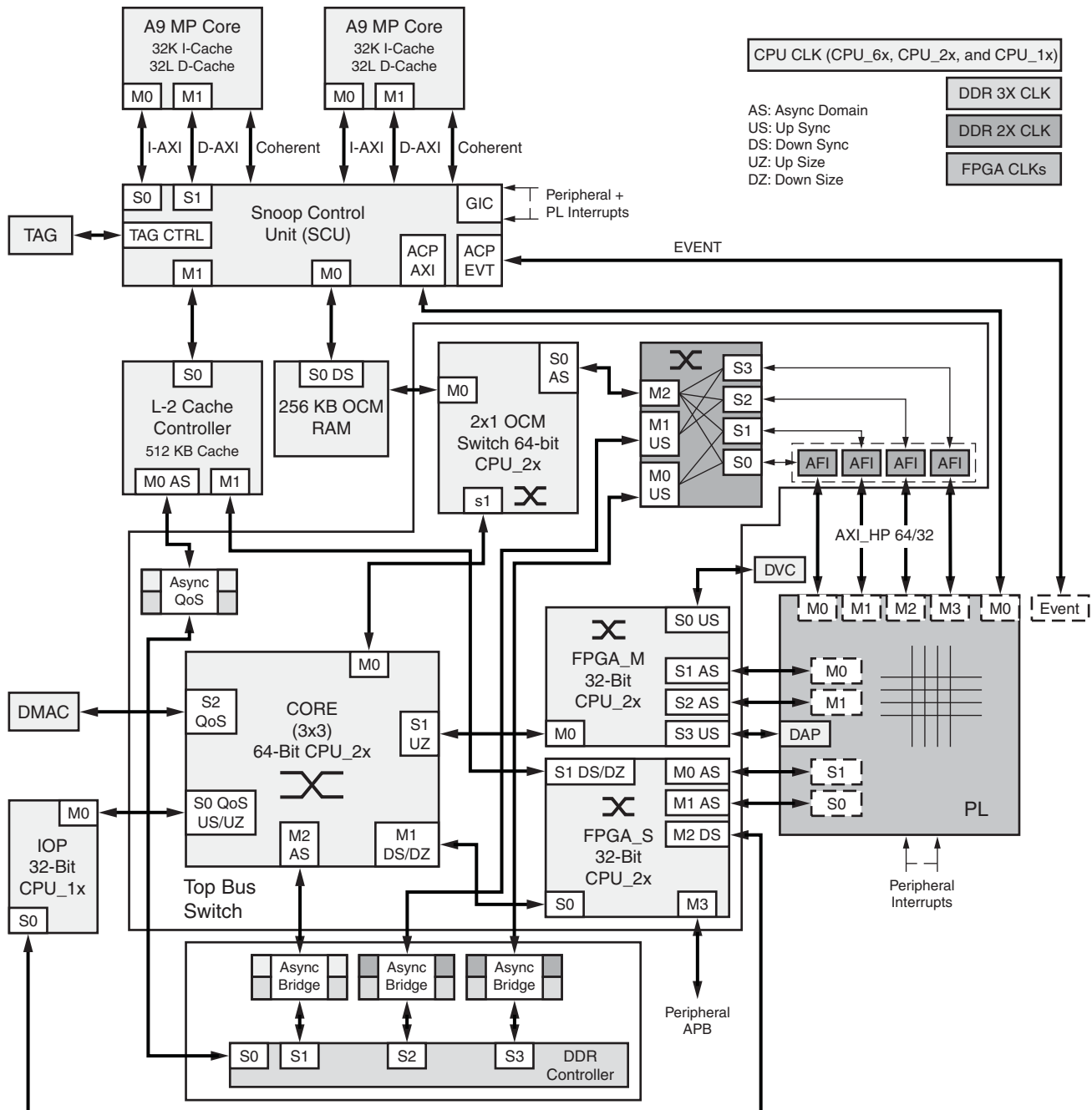
- Six-bit programmable frequency dividers
- Dynamic switching on most clock circuits
- Four clock generators for the PL

25.1.3 Reset

The clock subsystem is an integral part of the PS and is reset when the system is reset. When this occurs, all of the registers that control the clocking module return to their reset values.

25.1.4 System Viewpoint

Figure 25-1 shows the clock network and related domains from a system viewpoint.



UG585_c25_02_041612

Figure 25-2: System Clock Domains

A version of the CPU clock is used for most of the internal clocking. The DMA peripheral request interfaces and the PL AXI channels (AXI_HP, AXI_ACP and AXI_GP) have asynchronous interfaces between the PS and PL. The synchronization, where the clock domain crossing occurs, is located inside of the PS. Therefore, the PL provides the interface clock to the PS. Each of the aforementioned interfaces could use unique clocks in the PL.

25.2 Power Management

The overall approach to power management is described in [Chapter 24, Power Management](#). The clock generation subsystem facilitates clock disabling and frequency control which affects power consumption.

The PLL power consumption is directly related to the PLL output frequency. The power consumption can be reduced by using a lower PLL output frequency. Power can also be reduced if one or two of the PLLs are not required. For example, if all of the clock generators can be driven by the DDR PLL, then the ARM and I/O PLLs can be disabled to reduce power consumption. The DDR PLL is the only unit that can drive all of the clock generators.

Each clock can be individually disabled when not in use. In some cases, individual subsystems contain additional clock disabling and other power reduction features.

25.2.1 Top-Level (Central) Interconnect Clock Disable

The CPU clocks for the interconnect (CPU_2x and CPU_1x) can be stopped by setting the TOPSW_CLK_CTRL [0] bit to a 1. When this bit is set, the controller waits for the AXI interfaces to the L2-cache and SCU to become idle and for the FPGAIDLEN signal from the PL to assert. For the other interfaces, the system software must ensure that the interfaces are idle before disabling the interconnect clock. As soon as the PS detects traffic on the L2-cache or the SCU, or the FPGAIDLEN is deasserted, the clocks will be re-enabled.

25.3 CPU Clock Domains

Figure 25-3 shows the clock generation network in the CPU clock domains.

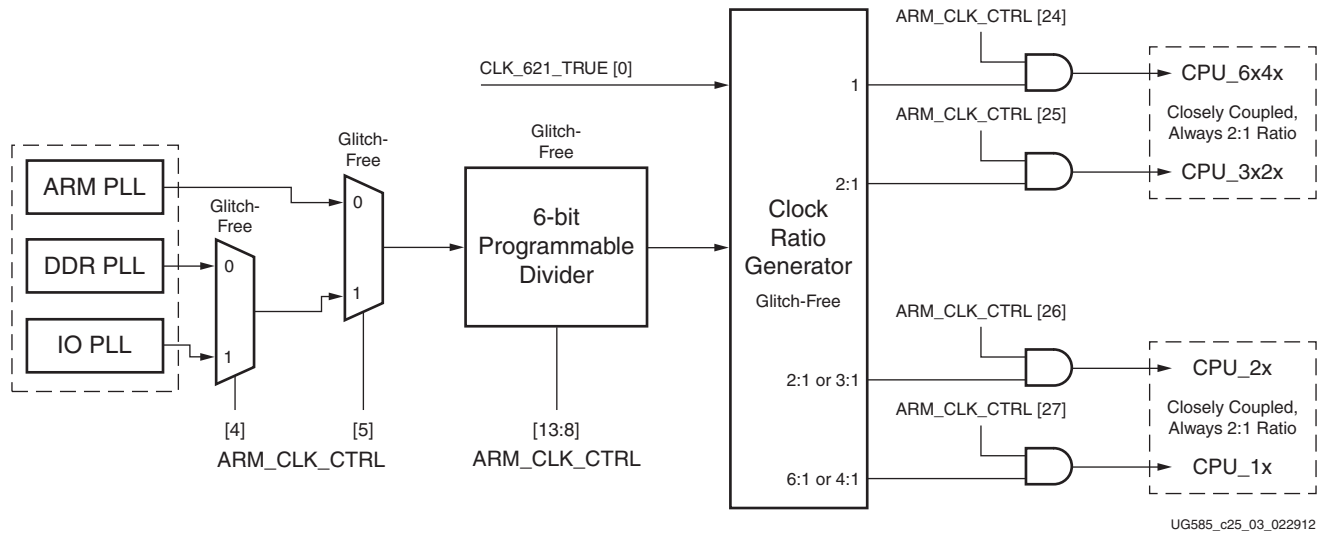


Figure 25-3: CPU Clock Generation and Domains

Ratio Examples

The clock frequencies listed in Table 25-1 are examples, see the applicable Zynq-7000 EPP data sheet for the specific allowed frequencies for each clock.

Table 25-1: CPU Clock Frequency Ratio Examples

CPU Clock	6:2:1	4:2:1	Clock Domain Description
CPU_6x4x	800 MHz (6 times faster than CPU_1x)	600 MHz (4 times faster than CPU_1x)	CPU clock frequency, CPU interconnect and OCM arbitration
CPU_3x2x	400 MHz (3 times faster than CPU_1x)	300 MHz (2 times faster than CPU_1x)	L2 Cache
CPU_2x	266 MHz (2 times faster than CPU_1x)	300 MHz (2 times faster than CPU_1x)	I/O peripheral AXI interconnect and OCM RAM
CPU_1x	133 MHz	150 MHz	I/O peripherals AHB and APB interface busses

Clock Usage

During normal usage, most system clocks will be derived by taking the input clock PS_CLK, sending it through the PLL, and finally dividing it down to be used within the PS. While the PS generates many different clocks, as shown in Figure 25-1, there are three clock domains that have the largest interaction and importance in the system: These are the DDR_3x domain, the DDR_2x domain, and the CPU clock domain. The DDR_3x clock domain includes the DDR memory controller. The DDR_2x

domain is primarily used for the high performance AXI interfaces to the PL (AXI_HP{0:3}) and the interconnect. The CPU clock domain controls the ARM processors along with many of the CPU peripherals.

The CPU clock domain is composed of four separate clocks: CPU_6x4x, CPU_3x2x, CPU_2x, and CPU_1x. These four clocks are named according to their frequencies, which are related by one of two ratios: 6:3:2:1 or 4:2:2:1 (abbreviated 6:2:1 and 4:2:1). The operating clock ratio is determined by the CLK_621_TRUE [0] bit value. In the 6:2:1 mode, the frequency of the CPU_6x4x clock is 6 times as fast as the CPU_1x clock. Table 1 shows examples of how these clocks are related. Refer to the applicable Zynq-700 EPP data sheet for the maximum clock frequency of each clock domain.

All the CPU clocks are synchronous to each other; while the DDR clocks are independent of each other and the CPU clocks. The I/O peripheral clocks, such as CAN reference clocks and SDIO reference clocks, are all generated by a similar method, starting from the PS_CLK pin, through a PLL, then a divider, and finally to the peripheral destination. Each peripheral clock is completely asynchronous to all other clocks.

Interconnect Clock Domains

The individual clock domains are shown in [Figure 25-2](#). The central interconnect has two main clock domains: the DDR_2x and the CPU_2x. For the five sub-switches, four clocks are in the CPU_2x clock domain, while the memory interconnect clock is in the DDR_2x clock domain. The direct path between the CPU (via the L2 cache) and DDR controller is in the DDR_3x clock domain, ensuring maximum throughput. The direct path between CPU and OCM is in the CPU_6x4x clock domain. The direct path between the SCU ACP and the PL is in the CPU_6x4x clock domain (clock domain crossing between the PL clock domain and the CPU clock domain is done in an asynchronous AXI bridge on the PS side of the PL).

CPU Clock Stop

Each CPU clock can be individually stopped using slcr.A9_CPU_RST_CTRL.A9_CLKSTOP{0,1}.

PS Peripheral Clocks

Every peripheral within the PS is supplied with its own independently gated version of the CPU clock. This gating is applied with a glitch-free clock gate, [Table 25-2](#).

Table 25-2: PS Peripheral Clock Control

AMBA Bus Peripheral	Base Clock	Control Bits in APER_CLK_CTRL	
		0: Disable 1: Enable	
DMAC	CPU_2x	DMA_CPU_2XCLKACT	[0]
USB 0	CPU_1x	USB0_CPU_1XCLKACT	[2]
USB 1	CPU_1x	USB1_CPU_1XCLKACT	[3]
GigE 0	CPU_1x	GEM0_CPU_1XCLKACT	[6]
GigE 1	CPU_1x	GEM1_CPU_1XCLKACT	[7]
SDIO 0	CPU_1x	SDIO_CPU_1XCLKACT	[10]

Table 25-2: PS Peripheral Clock Control (Cont'd)

AMBA Bus Peripheral	Base Clock	Control Bits in APER_CLK_CTRL 0: Disable 1: Enable	
SDIO 1	CPU_1x	SDI1_CPU_1XCLKACT	[11]
SPI 0	CPU_1x	SPI0_CPU_1XCLKACT	[14]
SPI 1	CPU_1x	SPI1_CPU_1XCLKACT	[15]
CAN 0	CPU_1x	CAN0_CPU_1XCLKACT	[16]
CAN 1	CPU_1x	CAN1_CPU_1XCLKACT	[17]
I2C 0	CPU_1x	I2C0_CPU_1XCLKACT	[18]
I2C 1	CPU_1x	I2C1_CPU_1XCLKACT	[19]
UART 0	CPU_1x	UART0_CPU_1XCLKACT	[20]
UART 1	CPU_1x	UART1_CPU_1XCLKACT	[21]
GPIO	CPU_1x	GPIO_CPU_1XCLKACT	[22]
Quad-SPI	CPU_1x	LQSPI_CPU_1XCLKACT	[23]
SMC	CPU_1x	SMC_CPU_1XCLKACT	[24]

System Performance

The clock frequencies of the different clock domains of the PS help dictate total system performance. In many cases, the highest frequency CPU clock results in the highest performance. However, some users will find that the CPU is not the critical performer in the system and that bandwidth across the interconnect is the bottleneck. In that case, it may be useful to switch the ratio from 6:2:1 to 4:2:1 mode. Depending on the device speed grade, the frequency of the CPU clocks may be limited by the cpu_6x while in 6:2:1 mode and may be limited by the cpu_2x clock while in 4:2:1 mode. Therefore, it is suggested that for those applications that might exchange some CPU performance for interconnect performance, check the appropriate data sheet to determine the optimal frequencies.

25.4 Clock Programming Examples

There are two clock configuration examples for 6:2:1 mode in Table 3. The first example is when the input PS_CLK is at 33.33 MHz and the second example is when the input frequency is at 50 MHz

The PLL output frequency is determined by the frequency of the input clock PS_CLK multiplied by the PLL Feedback Divider value (M value). In the example, below for the ARM PLL, $33.333 \times 40 = 1.333$ GHz. For each of the clocks listed in the lower half of the table, the clock frequency equals the sourced PLL frequency divided by the divisor value. For some of the peripheral clocks, such as CAN, Ethernet, and the PL, there are two cascaded dividers.

Table 25-3: Clock Frequency Setting Examples for 6:2:1 Mode

			Example 1			Example 2		
PS_CLK			33.333 MHz			50 MHz		
		PLL	PLL Feedback Divider Value		PLL Output Frequency	PLL Feedback Divider value		PLL Output Frequency
		ARM PLL	40		1333	20		1000
		DDR PLL	32		1067	16		800
		I/O PLL	30		1000	20		1000
Clock	No.	PLL Source	Divisor 0	Divisor 1	Clock Frequency	Divisor 0	Divisor 1	Clock Frequency
cpu_6x4x	1	ARM PLL	2	~ ⁽¹⁾	667	2	~	500
cpu_3x2x	1	ARM PLL		~	333		~	250
cpu_2x	1	ARM PLL		~	222		~	167
cpu_1x	1	ARM PLL		~	111		~	83
ddr_3x	1	DDR PLL	2	~	533	2	~	400
ddr_2x	1	DDR PLL	3	~	356	3	~	267
DDR DCI	1	DDR PLL	7	15	10	7	15	8
SMC	1	IO PLL	10	~	100	12	~	83
Quad SPI	1	IO PLL	5	~	200	6	~	167
GigE	2	IO PLL	8	1	125	8	1	125
SDIO	2	IO PLL	10	~	100	10	~	100
UART	1	IO PLL	40	~	25	40	~	25
SPI	2	IO PLL	5	~	200	8	~	125
CAN	2	IO PLL	10	1	100	12	1	83
PCAP	1	IO PLL	5	~	200	6	~	167
Trace Port	1	IO PLL	10	~	100	15	~	67
PLL FCLKs	4	IO PLL	20	1	50	20	1	50

1. "~" in the table indicates that there is no second divider (not applicable).

25.5 Basic Clock Branch Design

There are several different components to each clock generation circuit. This section describes a generic template that is used to explain the pieces used for all of the following I/O peripheral clocks. The most basic types include:

- 2-to-1 multiplexers for selecting a clock source
- Programmable divider(s)
- Glitch-free clock activation gate

These features are shown in [Figure 25-4](#).

Basic Clock Generator Design

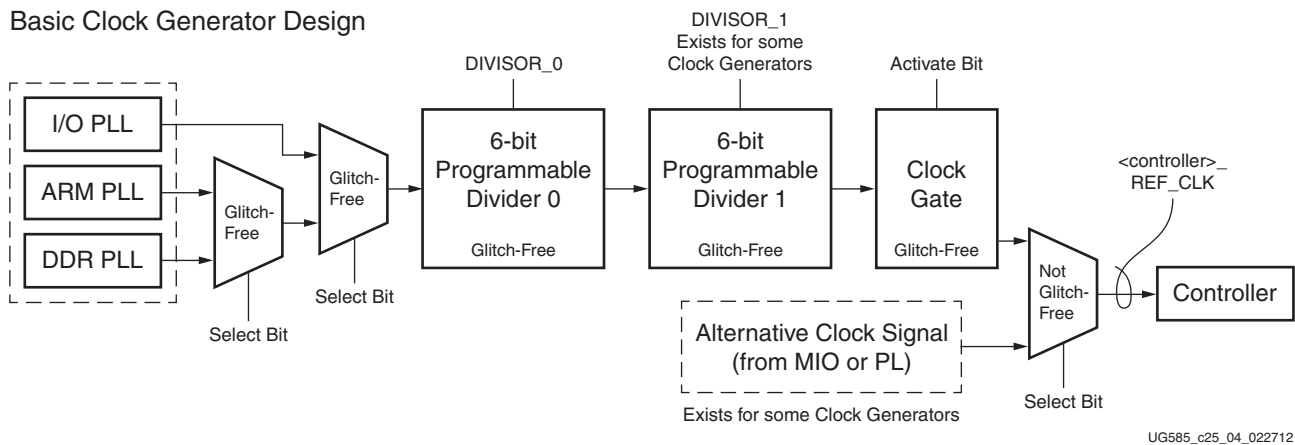


Figure 25-4: Basic Clock Branch Design

PLL

The PLL uses a feedback divider to create an output clock that is equal to the input reference clock multiplied by the PLL_FDIV value supplied by the SLCR.

Glitch-Free Clock Multiplexers

When a dynamic selection is required between two clock sources, the glitch-free multiplexers (GFM) change the clock selection on the low phase of the clock before starting the newly selected clock at the beginning of its low phase. The GFM operates properly only if both input clocks are active. Switching while either of the clocks is inactive causes the switching operation to fail.

Glitch-Free Clock Gate

The glitch-free clock gate is used when a dynamic gating is required to enable and disable a clock source. The gate ensures that the clock is terminated and re-enabled cleanly on its low phase.

Glitch-Free Divider

The glitch-free dividers take an input clock and divide it based on the divisor. When the divisor is changed, the output smoothly transitions to the new clock frequency without any glitches.

Clock Select Multiplexers

The clock source multiplexers select between the local clock generated by a clock generator and another source that is external to the clock generator. The clock source multiplexer is not glitch free.

25.6 DDR Clock Domains

There are two independent DDR clock domains: DDR_2x and DDR_3x. The DDR memory ports, controller and DRAM interface are clocked by DDR_3x (see [Figure 25-5](#)). The AXI_HP ports and the AXI_HP interconnect paths from the AXI_HP to the DDR Interconnect module are clocked by DDR_2x.

These clocks are asynchronous to each other and can have a wide range of frequency ratios between them.

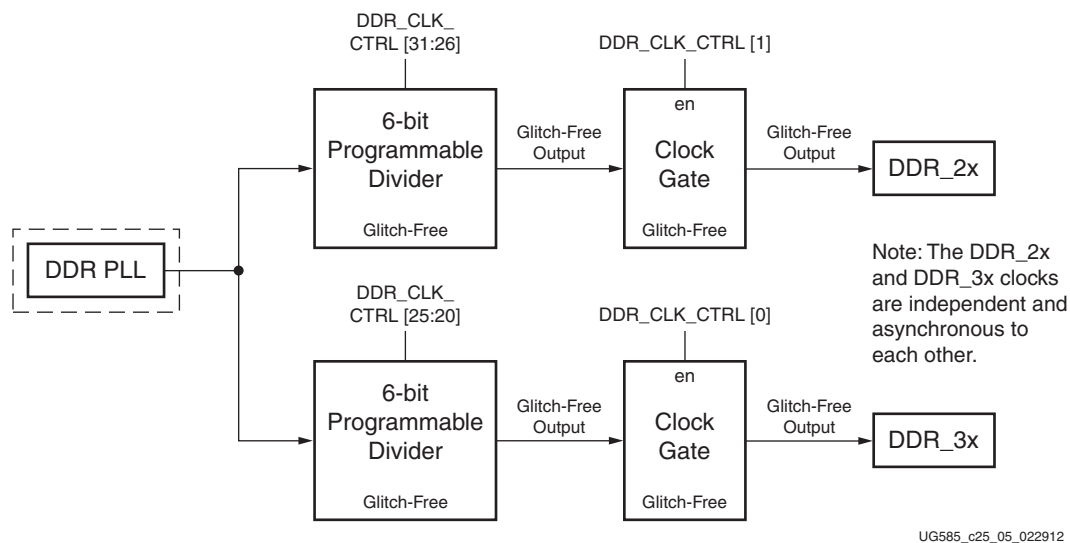


Figure 25-5: DDR Clock Generation

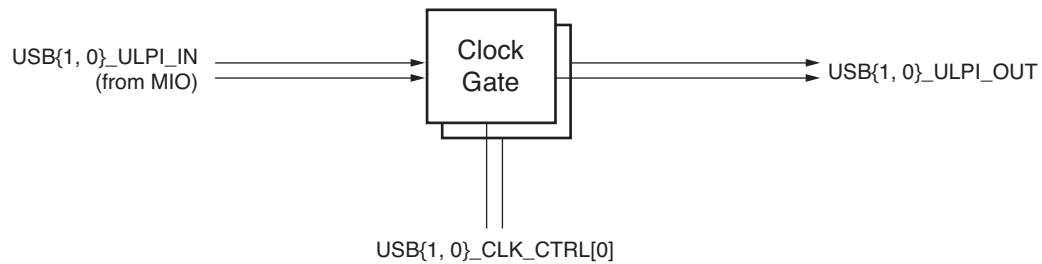
25.7 I/O Peripheral (IOP) Clocks

I/O peripheral (IOP) clocks:

- USB
- Ethernet
- SDIO
- SMC
- SPI
- Quad-SPI
- UART
- CAN

25.7.1 USB Clocks

The ULPI clock comes from a USB MIO pin. The ULPI clock is gated with an enable and is used to clock the interface side of the USB controller as shown in [Figure 25-6](#).



UG585_c25_06_022712

Figure 25-6: USB Clocks

25.7.2 Ethernet Clocks

The Ethernet Clocks generation network is shown in Figure 25-7.

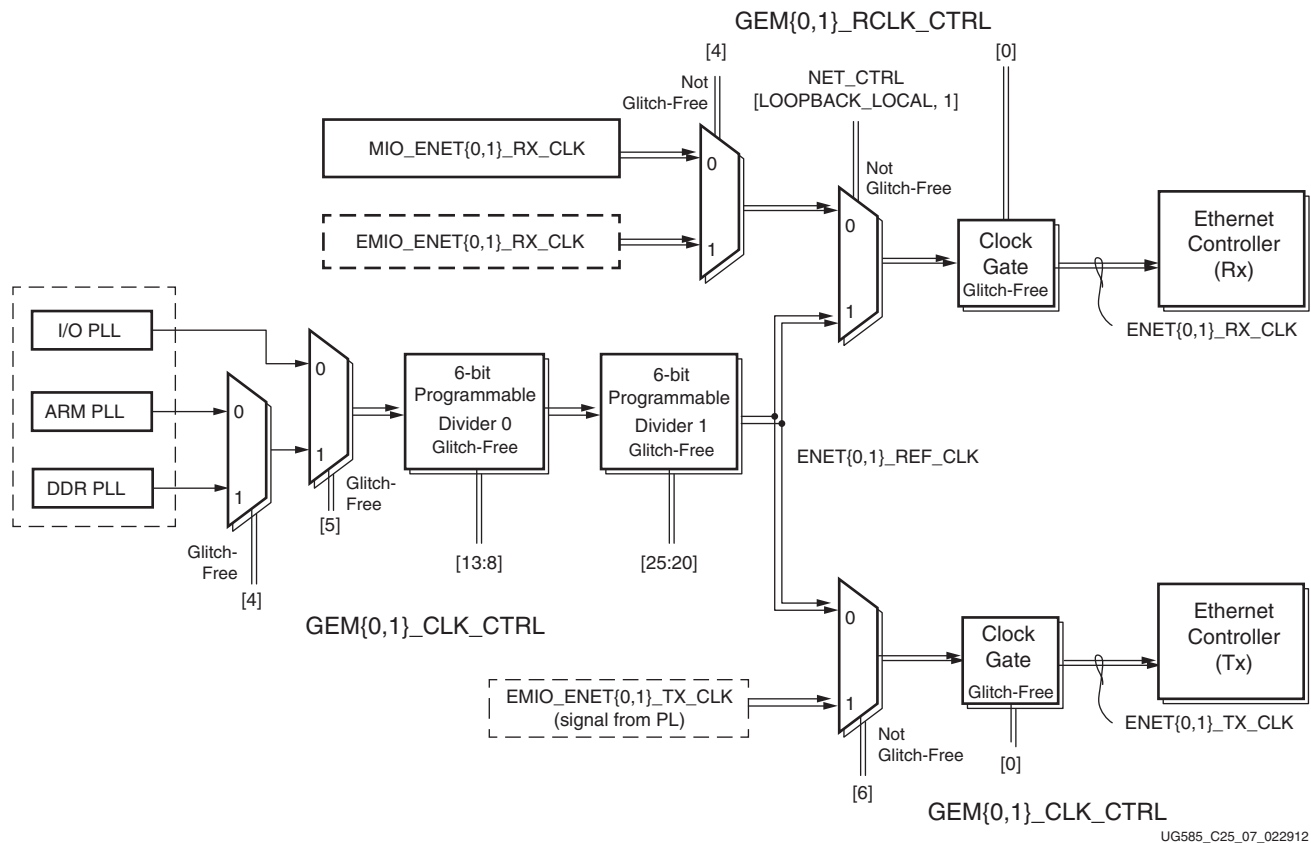


Figure 25-7: Ethernet Clock Generation

Ethernet Receiver Clocks

There are two Ethernet receiver clocks: gem0_rx_clk and gem1_rx_clk. In normal functional mode, these are either sourced from an external Ethernet PHY via the MIO or an extended MIO (EMIO). For MAC internal loopback mode, these clocks are sourced from the internal Ethernet reference clocks. They are also gated with an enable that can be used for power saving control. These are used to clock the receiver side of the Gigabit Ethernet MAC IP.

Neither the source selection nor the loopback selection multiplexers are glitch free because the source clocks might not be present. It is anticipated that clock gating is applied before any change in selection. To support loopback mode, gem0_rx_clk and gem1_rx_clk are supplied with gem0_ref_clk and gem1_ref_clk.

Ethernet Transmit Clocks

Two Ethernet clocks are required to be generated: gem0_ref_tx_clk and gem1_ref_tx_clk. These are used to clock the transmit side of the Ethernet MACs and as a source synchronous output clock for

the RGMII interface. They are also used to provide a stable reference clock to the Ethernet receive paths when internal loopback mode is selected.

These clocks can also be sourced from the EMIO. In this case, the associated RGMII interface is disabled and the MAC connects to the PL through an MII or GMII interface. In this case, the Ethernet reference clock must be provided by the PL. This is regardless of MII or GMII, where normally tx_clk is an input in MII and an output in GMII.

When operating in MII or GMII mode, its reference clock is provided by the PL through the eth*_emio_tx_clk. The EMIO source multiplexer is not glitch free because the EMIO source clock cannot be relied upon to be present. It is anticipated that this source selection is a static configuration or that the generated clock be gated before changing to the EMIO source. To support loopback mode, gem0_rx_clk and gem1_rx_clk are supplied with gem0_ref_clk and gem1_ref_clk.

25.7.3 SDIO, SMC, SPI, Quad-SPI and UART Clocks

The SDIO, SMC, Quad SPI, and UART peripheral clocks all have the same programming model (see [Figure 25-8](#)). The PLL source and divider values are shared for each I/O peripheral controller. The clocks for each SDIO, SPI and the UART controller can be individually enabled/disabled. There is a single clock, each, for the SMC and Quad SPI controllers.

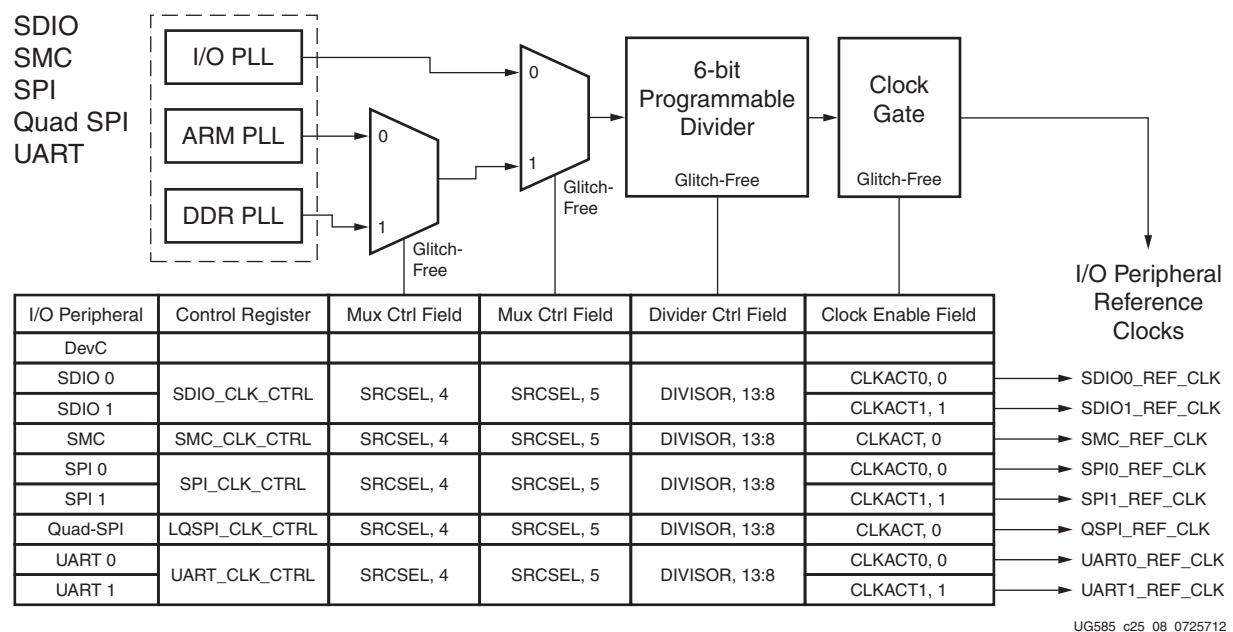


Figure 25-8: SDIO, SMC, SPI, Quad SPI and UART Reference Clocks

25.7.4 CAN Clocks

There are two controller area network (CAN) reference clocks: CAN0_REF_CLK and CAN1_REF_CLK. Both clocks share the same PLL source selection and dividers as shown in [Figure 25-9](#). Each clock has independent alternate source selection (MIO pin or the clock generator), and independent clock gates. These clocks are used for the I/O interface side of the CAN peripherals.

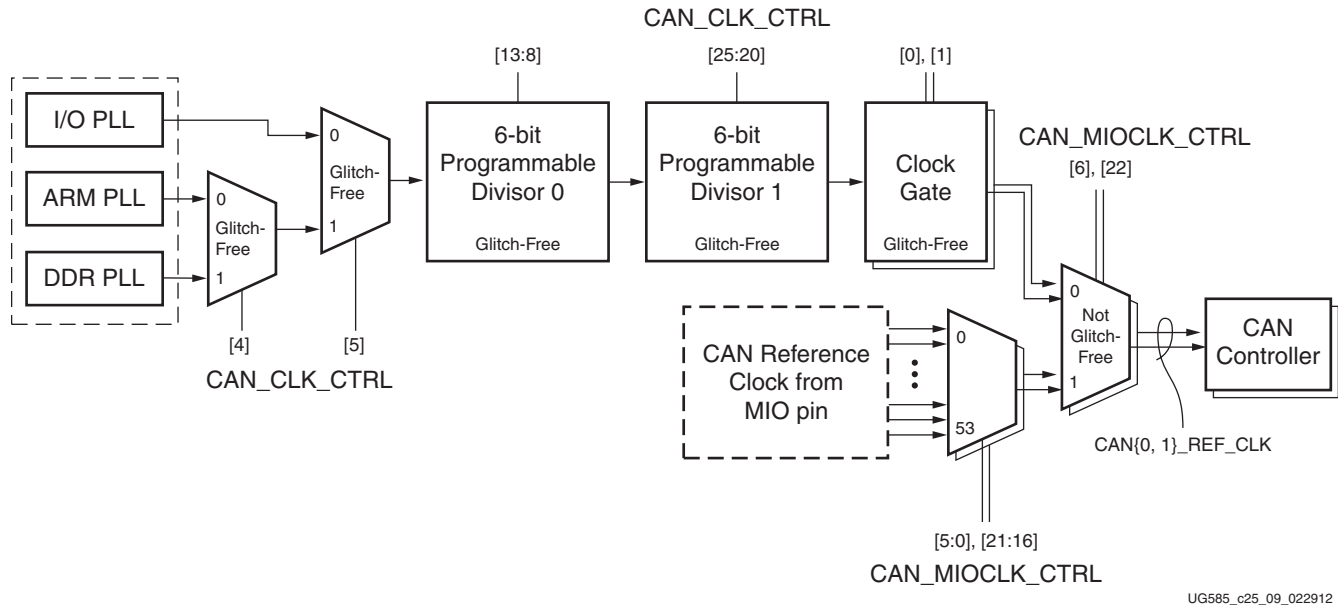


Figure 25-9: CAN Clock Generation

25.8 GPIO and I2C Clocks

The GPIO and I2C peripherals are clocked by the CPU_1x APB bus interface clock provided by the interconnect (refer to section [25.3 CPU Clock Domains](#)).

25.9 PL Clocks

The PL has its own clock management generation and distribution features and also receives four clock signals from the clock generator in the PS (see [Figure 25-9](#)). For the details of the PL clocking structures, see the 7 series FPGAs clocking documentation.

The four clocks that are generated by the PS are completely asynchronous to each other with no relationship to other PL clocks.

The four clocks are derived from individually selected PLLs in the PS. Each of the PL clocks are independent output signals that produce suitable clock waveforms for PL use.

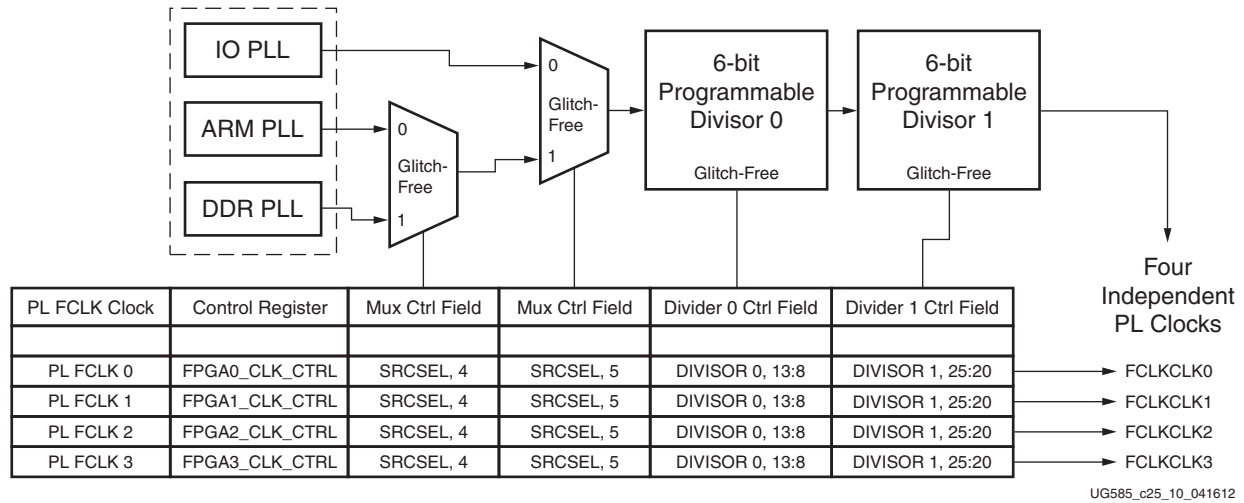


Figure 25-10: PL Clock Generation

25.10 Miscellaneous Clocks

25.10.1 Trace Port Clock

The trace port clock, is used to clock the top-level trace port and trace buffer within the SoC debug. This generated clock must be twice the frequency of the desired trace port clock. The frequency of trace_clk must be chosen to be fast enough to allow the trace port to keep up with the amount of data being traced, but slow enough to meet the dynamic characteristics of the output buffer. To allow some flexibility, TRACE_CLK is internally generated from a divided PLL output (see Figure 25-11).

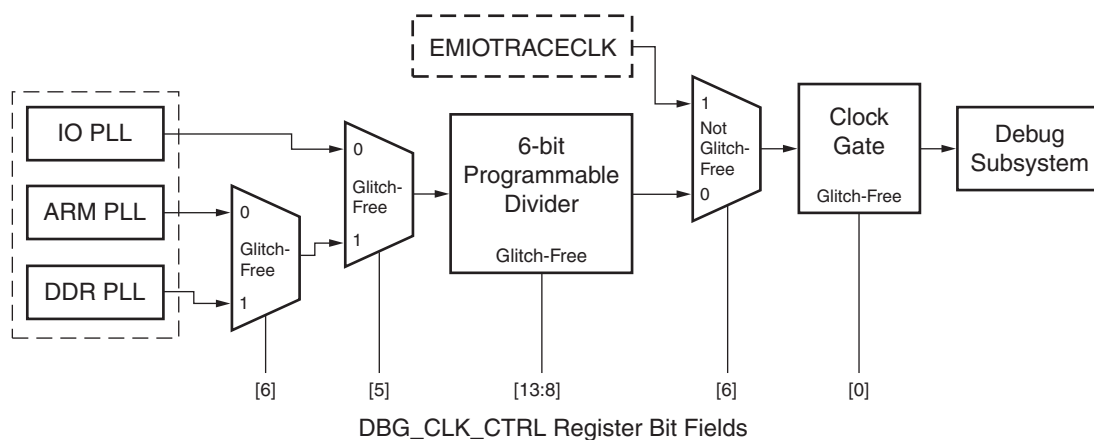


Figure 25-11: Trace Port Clock Generation

25.11 Register Overview

An overview of the PS clock subsystem registers is shown in [Table 25-4](#).

Table 25-4: Clock Generation Register Overview

Register	Description	Comments
PLL		
PLL_STATUS	PLL status	
ARM_PLL_CTRL	CPU PLL control	<ul style="list-style-type: none">Control registers include:<ul style="list-style-type: none">Reset, power-down, bypassDivisor(s)Configuration registers include:<ul style="list-style-type: none">PLL control parameters (see Table 25-5)
ARM_PLL_CFG	CPU PLL configuration	
DDR_PLL_CTRL	DDR PLL control	
DDR_PLL_CFG	DDR PLL configuration	
IO_PLL_CTRL	I/O PLL control	
IO_PLL_CFG	I/O PLL configuration	
CPU, DDR and Interconnect		
ARM_CKL_CTRL	CPU clock control	
CLK_621_TRUE	Select CPU clock frequency ratio	6:2:1 or 4:2:1
DDR_CLK_CTRL	DDR clock control	
APER_CLK_CTRL	AMBA peripheral clock control	
TOPSW_CLK_CTRL	Top-level switch clock control	
PL Clocks		
FPGA{3:0}_CLK_CTRL	PS to PL output clock control	
FPGA{3:0}_THR_{CTRL, CNT, STA}	PS to PL output clock throttle control, count and status	
I/O Peripheral Clocks		
USB{1, 0}_CLK_CTRL	USB ULPI clock control	<ul style="list-style-type: none">DIVISOR bit field (one or two parameters, depending on the peripheral)Clock enable active control
GEM{1, 0}_CLK_CTRL	Gigabit Ethernet Rx clock control	
GEM{1, 0}_CLK_CTRL	Gigabit Ethernet ref clock control	
SMC_CLK_CTRL	SMC ref clock control	
LQSPI_CLK_CTRL	Quad-SPI ref clock control	
SDIO_CLK_CTRL	SDIO ref clock control	
UART_CLK_CTRL	UART ref clock control	
SPI_CLK_CTRL	SPI ref clock control	
CAN_CLK_CTRL	CAN ref clock control	
CAN_MIOCLK_CTRL	CAN com port clock control	
System Debug Clocks		
DBG_CLK_CTRL	CoreSight SoC trace clk control	
PCAP_CLK_CTRL	PCAP clock control	

25.12 Programming Model

25.12.1 Branch Clock Generator

Each clock generator has a clock control register `<module>_CLK_CTRL`. Within the clock control register, the SRCSEL field selects a clock source and the DIVISOR field is the amount the source clock is divided down to produce the desired clock frequency. Some clock generators have two cascaded dividers.

To prevent the clock generator from exceeding the maximum frequency of the attached subsystem, program the SRCSEL and the DIVISOR values in three separate steps:

1. Increase the value of the DIVISOR, as needed, so that the two PLL clock sources will not cause the clock generator to exceed the maximum clock frequency of the subsystem.
2. Set the SRCSEL to the desired source.
3. Update the DIVISOR to the desired value.

6-bit Programmable Divider

The 6-bit divider provides a divide range of 1 to 63, supports both even and odd divide values while producing a close to 50% duty cycle, and is glitchless (divide values can be modified dynamically). The only exception to this rule is that the DDR_3X divider can only be programmed to divide by an even divisor. Some reference clocks have one divider and some have two dividers.

25.12.2 DDR Clocks

It is a requirement that the DDR_3x clock must always be programmed to an even divisor.

Note that changing the divider frequency does not produce glitches on the clock, however the DDR controller will not necessarily operate correctly if the frequency is changed without modifying its timing parameters. Therefore, it is suggested to first idle the DDR controller when the DDR clock is to be reprogrammed.

25.12.3 Digitally Controlled Impedance (DCI) Clock

The digitally controlled impedance (DCI) clock is required by the DDR PHY for calibration. The DCI clock is a low frequency clock, normally set to 10 MHz.

25.12.4 PLLs

The three PLLs share the clock input signal, PS_CLK. Each PLL can be bypassed individually under software control. As part of the power-on reset sequence, all PLLs can be bypassed using the pll_bypass boot mode pin straps. (Refer to the boot mode section of [Chapter 6, Boot and Configuration](#).)

The PLL configuration and control registers are in the SLCR. The PLL frequency control registers include the M (feedback divide ratio also known as PLL_FDIV), LOCK_CNT, PLL_CP, and PLL_RES control fields. For each divide ratio M, the PLL_CP, PLL_RES, and LOCK_CNT fields must also be updated to the values shown in Table 25-5. The worst case lock time for the PLLs is 60 us.

Enable PLL Mode when PLL Bypass Mode Pin is Tied High

After the system boots in bypass mode, the following sequence can be used to enable a PLL. Each PLL can be individually enabled. This example shows how to enable the ARM PLL:

1. Program the ARM_PLL_CTRL[PLL_FDIV] value and the PLL configuration register, ARM_PLL_CFG[LOCK_CNT, PLL_CP, PLL_RES], to their required values.
2. Force the PLL into bypass mode by writing a 1 to ARM_PLL_CTRL [PLL_BYPASS_FORCE, 4] and setting the ARM_PLL_CTRL [PLL_BYPASS_QUAL, 3] bit to a 0. This de-asserts the reset to the ARM PLL.
3. Assert and de-assert the PLL reset by writing a 1 and then a 0 to ARM_PLL_CTRL [PLL_RESET, 0].
4. Verify that the PLL is locked by reading PLL_STATUS [ARM_PLL_LOCK, 3].
5. Disable the PLL bypass mode by writing a 0 to ARM_PLL_CTRL [4].

The DDR and I/O PLLs are programmed in a similar fashion.

Software-Controlled PLL Update

The following steps are required for software to update the PLL clock frequency. This example is for the I/O PLL.

1. Program the IO_PLL_CTRL[PLL_FDIV] value and the PLL configuration register, IO_PLL_CFG [LOCK_CNT, PLL_CP, PLL_RES].
2. Force the PLL into bypass mode by writing a 1 to IO_PLL_CTRL [PLL_BYPASS_FORCE, 4].
3. Assert and de-assert the PLL reset by writing 1 and then a 0 to IO_PLL_CTRL [PLL_RESET, 0].
4. Verify that the PLL is locked by reading PLL_STATUS [IO_PLL_LOCK, 2].
5. Disable the PLL bypass mode by writing a 0 to IO_PLL_CTRL [4].

Table 25-5: PLL Frequency Control Settings

Desired PLL Multiplier	PLL Control and Configuration Bit Fields			
	PLL_FDIV	PLL CP	PLL RES	LOCK CNT
13	13	2	6	750
14	14	2	6	700
15	15	2	6	650
16	16	2	10	625
17	17	2	10	575
18	18	2	10	550
19	19	2	10	525
20	20	2	12	500

Table 25-5: PLL Frequency Control Settings (Cont'd)

Desired PLL Multiplier	PLL Control and Configuration Bit Fields			
	PLL_FDIV	PLL CP	PLL RES	LOCK CNT
21	21	2	12	475
22	22	2	12	450
23	23	2	12	425
24 ~ 25	24 ~ 25	2	12	400
26	26	2	12	375
27 ~ 28	27 ~ 28	2	12	350
29 ~ 30	29 ~ 30	2	12	325
31 ~ 33	31 ~ 33	2	2	300
34 ~ 36	34 ~ 36	2	2	275
37 ~ 40	37 ~ 40	2	2	250
41 ~ 47	41 ~ 47	3	12	250
48 ~ 66	48 ~ 66	2	4	250

25.12.5 CPU_DIVISOR(ARM_CLK_CTRL[13:8])

To improve the quality of the high speed clocks going to the CPU and DDR, there is a requirement that they get divided by an even number by the divisor described above. For the DDR, the odd divisor has already been removed and is never legal. For the cpu_divisor, users must provide slightly more flexibility in that they can program the following values: $x=2$, $x=4$, $x>4$. The odd divisors were left because at low frequency it is not a requirement to divide by an even number.

Reset System

26.1 Introduction

The reset system includes resets generated by hardware, watchdog timers, the JTAG controller, and software. Every module and system in Zynq-7000 EPP devices includes a reset that is driven by the reset system. Hardware resets are driven by the power-on reset signal (PS_POR_B) and the system reset signal (PS_SRST_B).

There are three watchdog timers in the PS that can generate resets. The JTAG controller can generate a reset that only resets the debug portion of the PS and a system-level reset. Software can generate individual sub-module resets or a system-level reset.

Resets are caused by many different sources and go to many different destinations. This chapter identifies all of the resets and either explains their functionality or provides a pointer to another chapter or another document.

26.1.1 Features

The key features of the reset system:

- Collects resets from hardware, watchdog timers, the JTAG controller and software
- Drives the reset of every module and subsystem
- Is an integral part of the device security system
- Executes a three-stage sequence: power-on, memory clear, and system enabling

26.1.2 Block Diagram

Figure 26-1 illustrates the logical dependencies of the main types of generated resets upon these reset triggers. This block diagram is intended to highlight dependencies, and does not accurately depict implementation detail or sequence timing.

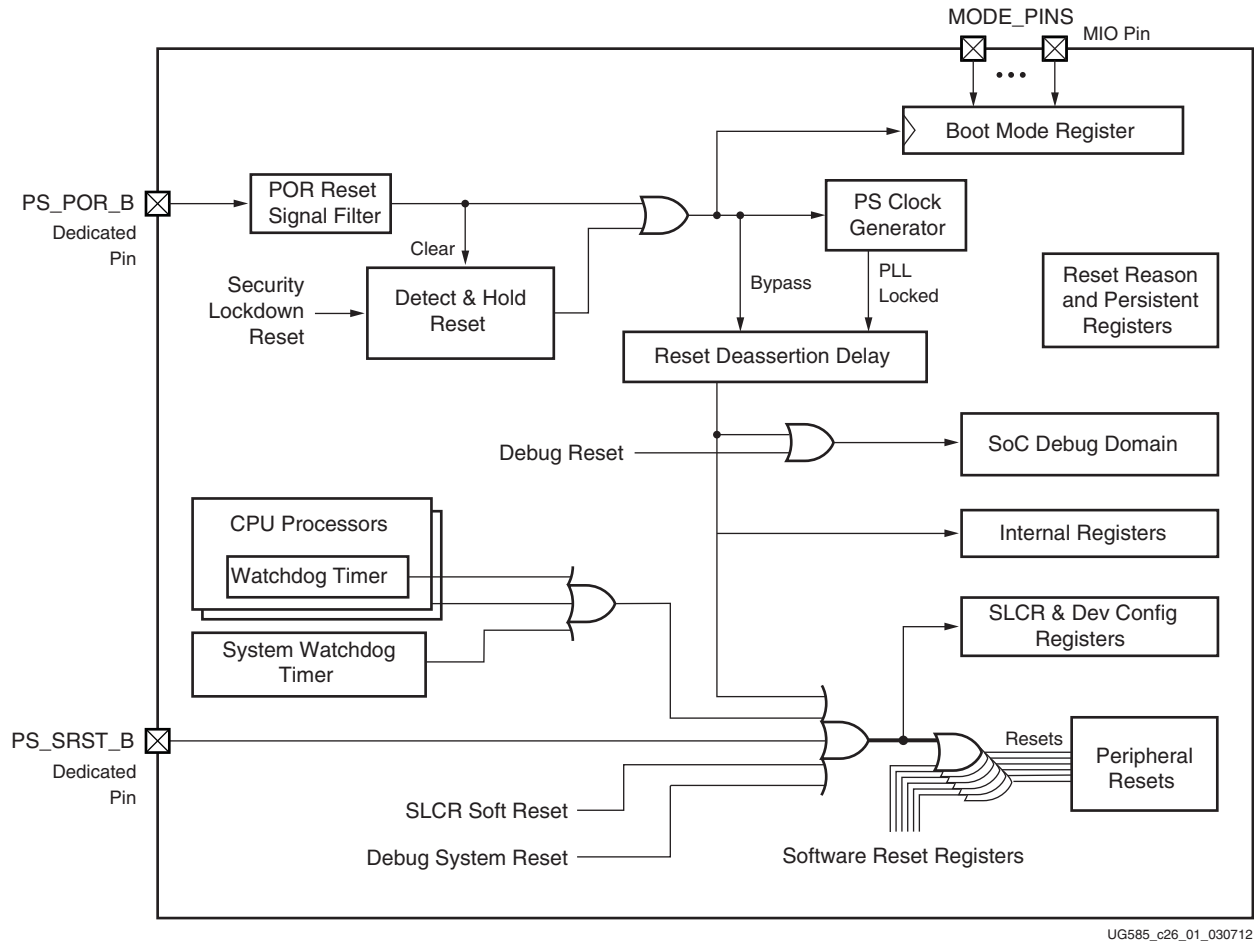


Figure 26-1: Resets Block Diagram

26.1.3 Reset Hierarchy

There are many different types of resets within the PS, from power-on reset that resets the entire system, to a peripheral reset which resets a single subsystem under software control. Figure 26-2 shows the relationships between all major reset signals within the PS. Reset signals flow from the top downwards. For example, power-on reset (POR) resets all logic within the PS, but system reset only resets the functions indicated in the diagram. This diagram presents the reset hierarchy of operation rather than the reset signal routing shown in Figure 26-1.

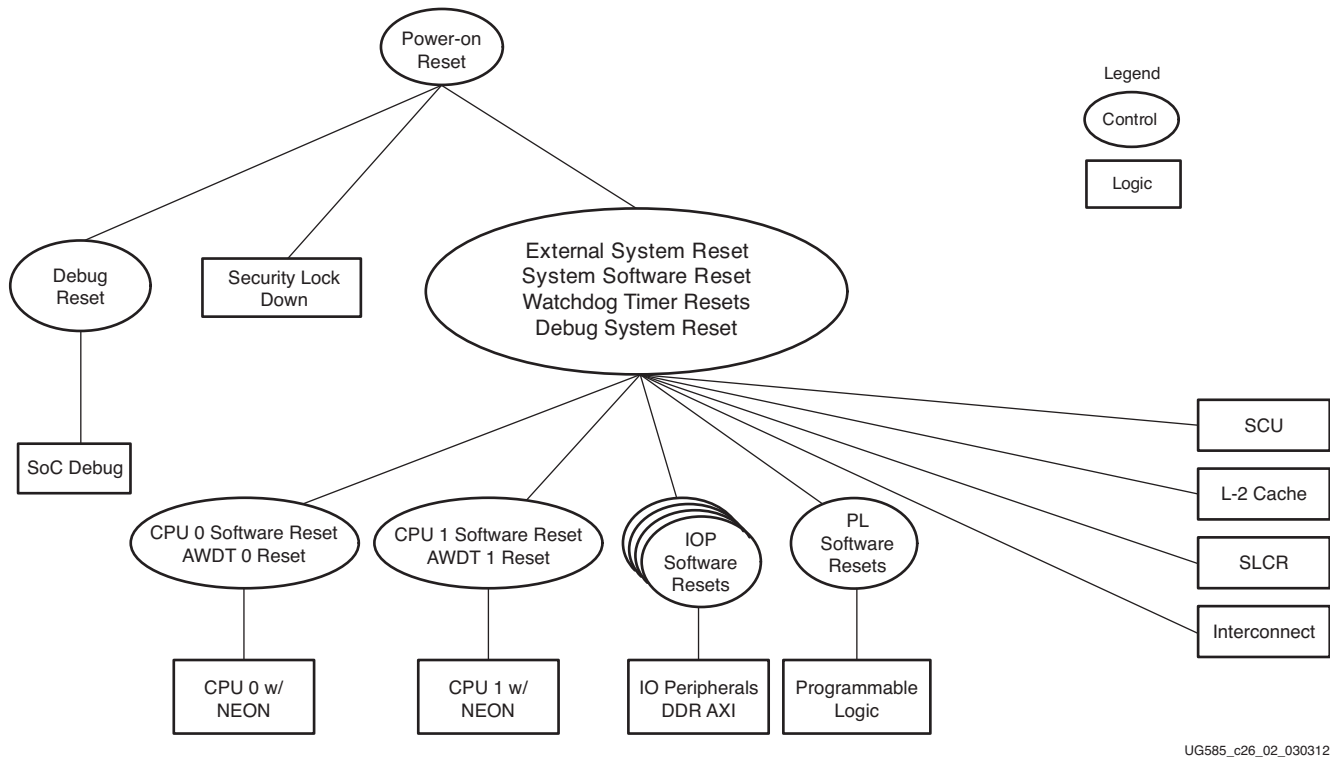


Figure 26-2: Reset Hierarchy Diagram

26.1.4 Boot Flow

The complete reset sequence is shown as in Figure 26-3. The first two steps are controlled by the external system and PS logic only starts to respond when power on reset (POR) is de-asserted. When the PS is operational, any type of reset can occur after POR. Those resets would insert into the flow chart, at their respective locations.

The POR signal can be asserted or de-asserted asynchronously. When the POR signal is de-asserted, it is conditioned to allow it to propagate cleanly to the clock module input logic and, if enabled, to the PLL clock circuits.

There is a BOOT_MODE strapping pin to select between all PLLs enabled and all PLLs disabled (bypassed).

The eFUSE controller comes out of reset. It automatically applies some data to the PLL, which allows for improved performance, and provides redundancy information to some of the RAM's in the PS. This activity is not visible to the user and cannot be affected by the user. This activity requires from 50 us to 100 us of time to complete but is otherwise not visible to the user.

If the PLLs are enabled, then the POR signal is delayed at this point until the PLL clocks have locked. The PLL clocks take up to 60 uS to lock. If PLL bypass mode is selected, the POR signal is not delayed.

Before the BootROM starts executing, the internal RAMs are cleared by hardware writing zeros into all addresses. For a listing of the times it takes to go through these steps, see section 6.3.3 [Boot Performance](#) in [Chapter 6, Boot and Configuration](#).

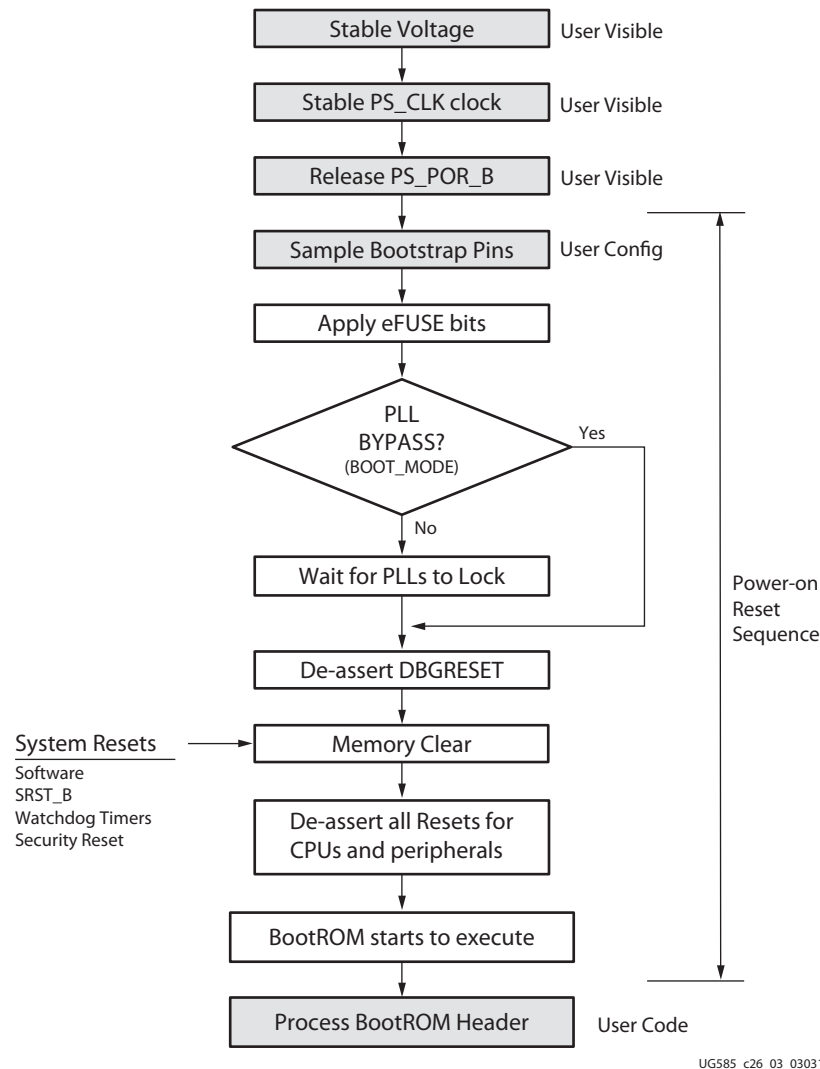


Figure 26-3: Power-On Reset Flow Diagram

26.2 Reset Sources

26.2.1 Power-on Reset (PS_POR_B)

The PS supports external power-on reset signals. The power-on reset is the master reset of the entire chip. This signal resets every register in the device capable of being reset.

The PS_POR_B reset pin is held Low until all PS power supplies are at their required voltage levels and PS_CLK is active. It can be asynchronously asserted and is internally synchronized and filtered. The filter prevents High-going glitches from entering the PS while the signal is intended to be held Low. It does not filter Low-going glitches when the signal is intended to be held high. Any Low-going glitch that is detected causes an immediate reset of the device.

The PS_POR_B signal is often connected to the power-good signal from the power supply. When PS_POR_B is de-asserted, the system samples the boot strap mode pins and begins its internal initialization process.

26.2.2 External System Reset (PS_SRST_B)

Power-on reset erases all debug configurations. The external system reset allows the user to reset all of the functional logic within the device without disturbing the debug environment. For example, the previous break points set by the user remain valid after system reset.

Due to security concerns, system reset erases all memory content within the PS, including the OCM. The PL is also reset in system reset. System reset does not re-sample the boot mode strapping pins.

If this pin is not used in the system, it should be tied high.

26.2.3 System Software Reset

The user can reset the entire system by asserting a software reset. By asserting, PSS_RST_CTRL[SOFT_RST], the entire system is reset with the same end result as the user pressing the PS_SRST_B pin (other than the RESET_REASON register value being different).

Just like the other system resets, all of the RAMs are cleared and the PL is reset as well.

26.2.4 Watchdog Timer Resets

The watchdog timer resets are internally generated by the watchdog timers when they are enabled and the timer expires. There are three different watchdog timers in the PS: one system-level timer (SWDT) and one private timer in each of the two ARM cores (AWDT0 and AWDT1). The system-level timer reset signal always resets the entire system, while the private watchdog timers can either reset just the ARM core that housed it, or the entire system.

26.2.5 Secure Violation Lock Down

When a security violation is detected, the entire PS is reset and locked down. After a security lock down occurs, the PS only becomes active again by asserting and de-asserting the PS_POR_B reset pin. Refer to [Chapter 32, Device Secure Boot](#) for details of how and when this signal is asserted.

26.2.6 Debug Resets

There are two types of debug resets that originate from the DAP controller; debug system reset and debug reset.

Debug system reset is a command from the ARM debug access port (DAP) which is controlled by JTAG. This causes the system to reset, just like the external system reset.

Debug reset resets certain portions of the SoC debug block including the JTAG logic.

The PS does not support the external TRST, although it does support assertion of a reset sequence using TMS. The JTAG logic is only reset at power-on reset or assertion of CDBG_RSTREQ from the ARM debug access port (DAP) Controller (JTAG). All of the logic in the JTAG TCK clock domain is reset by this signal.

26.3 Reset Effects

The effects of various resets are described in [Table 26-1](#).

Table 26-1: Reset Effects

Reset Name	Source	System	RAMs Cleared
Power-On Reset (PS_POR_B)	Device pin	Entire chip, including debug. (All) The PL must be re-programmed.	All
Security Lock Down (requires a power-on reset to recover)	DVI		All
System Reset (PS_SRST_B)	Device pin	All except debug and persistent registers. The PL must be re-programmed.	All
System Software	SLCR		All
System Debug Reset	JTAG		All
System Watchdog Timer	SWDT		All
CPU Watchdog Timers (when slcr.RS_AWDT_CTRL{1,0} = 0)	AWDT		All
CPU Watchdog Timers (when slcr.RS_AWDT_CTRL{1,0} = 1)	AWDT	CPU (s) only.	None
Debug Reset	JTAG	Debug logic.	None
Peripherals	SLCR	Selected peripherals.	None

26.3.1 Peripherals

All PS peripherals will be reset when the PS is reset. In addition, individual peripheral resets may also be asserted under software control, through programmable bits within the SLCR.

Most peripherals have the ability to reset each of the clock domains within that peripheral. For instance, the Ethernet controller can reset the RX side, TX side, or the interconnect side. Each clock domain can be reset separately. Please note, that individual peripherals may have their own resets defined within those blocks.

Peripheral resets will not result in the MBIST clear logic being activated to clear all memories within the design.

26.4 PL Resets

26.4.1 PL General Purpose User Resets

There are four separate reset signals routed to the PL which could be used as general purpose reset signals for PL logic. These reset signals are not removed until the PS is out of its boot sequence and user code de-asserts them. They are controllable by the SLCR.

26.5 Register Overviews

The following sections provide an overview of the reset control registers.

26.5.1 Reset Subsystem Control and Status

The Reset Subsystem control and Status registers are identified in [Table 26-2](#).

Table 26-2: Reset Subsystem Control and Status

Name	Description	HW Register Name
Reason	Remains persistent except by PS_POR_B	slcr.RST_REASON
Reason Clear	Clears the Reason register	slcr.RST_REASON_CLR
Reboot Status	Remains persistent except by PS_POR_B	slcr.REBOOT_STATUS
Persistent	Remains persistent except by PS_POR_B	devcfg.LOCK, devcfg.MULTIBOOT_ADDR, devcfg.STATUS, devcfg.UNLOCK, slcr.SLCR.REBOOT_STATUS, slcr.RESET_REASON, DEVCFG.CTRL.MULTIBOOT_EN, DEVCFG.CTRL.PCFG_AES_FUSE, DEVCFG.CTRL.PCFG_AES_EN, DEVCFG.CTRL.SEU_EN, DEVCFG.CTRL.SEC_EN, DEVCFG.LOCK, DEVCFG.STATUS.SECURE_RST, SLCR.ARM_CLK_CTRL.SRCSEL, SLCR.APU_CTRL.CFGSDISABLE, SLCR.APU_CTRL.CP15SDISABLE, SCU.Watchdog_Reset_Status_Register
Debug		See Chapter 28, System Test and Debug

The persistent registers are registers that are not cleared during any reset other than the POR reset. Two types of persistent registers are used; one stores the state for user identification, and the other is used for security reasons. For information on the security registers, please see [Appendix B, Register Details](#), or [Chapter 32, Device Secure Boot](#).

The first group includes the RST_REASON and the REBOOT_STATUS registers. The RST_REASON register indicates the cause of the last reset. User code should read this register, act on it, then clear it out using the slcr.RST_REASON_CLR register so it is primed for the next event. This register is written to by hardware. REBOOT_STATUS is written to by the BootROM. The details of values in the REBOOT_STATUS register can be found in [Chapter 6, Boot and Configuration](#).

26.5.2 System Reset Control

System Reset registers are identified in [Table 26-3](#).

Table 26-3: System Reset Control

Name	Systems Affected	HW Register Name
System Software Reset	All	slcr.PSS_RST_CTRL

26.5.3 Peripheral Reset Control

Peripheral Reset registers are identified in [Table 26-4](#).

Table 26-4: System Resets Register Overview

Peripheral Name	Description	HW Register
AXI Interconnect	Top level interconnect domain	slcr.TOPSW_RST_CTRL
CPU and Peripherals	CPUs, L1/MMU, SCU, WDT, Timer	slcr.A9_CPU_RST_CTRL
OCM	On chip memory	slcr.OCM_RST_CTRL
CAN	CPU 1x	slcr.CAN_RST_CTRL
DDR	DDR PHY/controller/control registers	slcr.DDR_RST_CTRL
DMA	DMA interface	slcr.DMAC_RST_CTRL
DevC	Subsystem, PCAP 2x and CPU 1x	slcr.DEVCI_RST_CTRL
Ethernet	Ref, Rx and CPU 1x	slcr.GEM_RST_CTRL
PL Fabric	PL, DMA req/ack, AXI interfaces	slcr.FPGA_RST_CTRL
GPIO	CPU 1x	slcr.GPIO_RST_CTRL
I2C	CPU 1x	slcr.I2C_RST_CTRL
Quad-SPI	Ref and CPU 1x	slcr.LQSPI_RST_CTRL
SDIO	Ref and CPU 1x	slcr.SDIO_RST_CTRL
SPI	Ref and CPU 1x	slcr.SPI_RST_CTRL
SMC	Ref and CPU 1x	slcr.SMC_RST_CTRL
UART	Ref and CPU 1x	slcr.UART_RST_CTRL
USB	ULPI and CPU 1x	slcr.USB_RST_CTRL

JTAG and DAP Subsystem

27.1 Introduction

The Zynq-7000 family of EPP devices provides debug access via a standard JTAG (IEEE 1149.1) debug interface. Internally, the EPP device implements both an ARM debug access port (DAP) inside the Processing System (PS) as well as a standard JTAG test access port (TAP) controller inside the Programmable Logic (PL). The ARM DAP as part of ARM CoreSight debug architecture allows the user to leverage industry standard third-party debug tools.

In addition to the standard JTAG functionality, the Xilinx tap controller supports a number of PL features including PL Debug, eFuse/BBRAM programming, on-chip XADC access, etc. Most importantly, it also allows debugging of ARM software through the DAP and PL hardware by using the TAP simultaneously with shared a trace buffer and cross triggering interface between the PS and PL.

Another important debug feature the Zynq-7000 EPP includes is debug trace support. This feature allows the user to capture both the PS and PL trace into a common trace buffer that is either read out through JTAG, described below, or sent out through the trace port interface unit (TPIU), described in [Chapter 28, System Test and Debug](#).

27.1.1 Block Diagram

[Figure 27-1](#) shows the top level DAP/TAP architecture. Based on the JTAG BOOT_MODE pin setting, the controllers will be in either cascaded JTAG chain or independent modes. As soon as the BootROM passes control to user software, the JTAG chain is enabled automatically, assuming a non-secure boot process. This allows debugging from the user software entry point.

JTAG supports two different modes: cascaded JTAG chain mode (also referred to as single chain mode) and independent JTAG chain mode (also referred to as split chain mode). The mode is determined through the mode input when the system comes out of reset.

In cascaded JTAG chain mode, which is controlled by the mode pin at the PS, both the TAP and DAP will be visible from external JTAG debug tools or a JTAG tester. With dedicated PL_TDO/TMS/TCK/TDI I/O at the PL side, only one JTAG cable can be connected, though it has access both to PS and PL features concurrently.

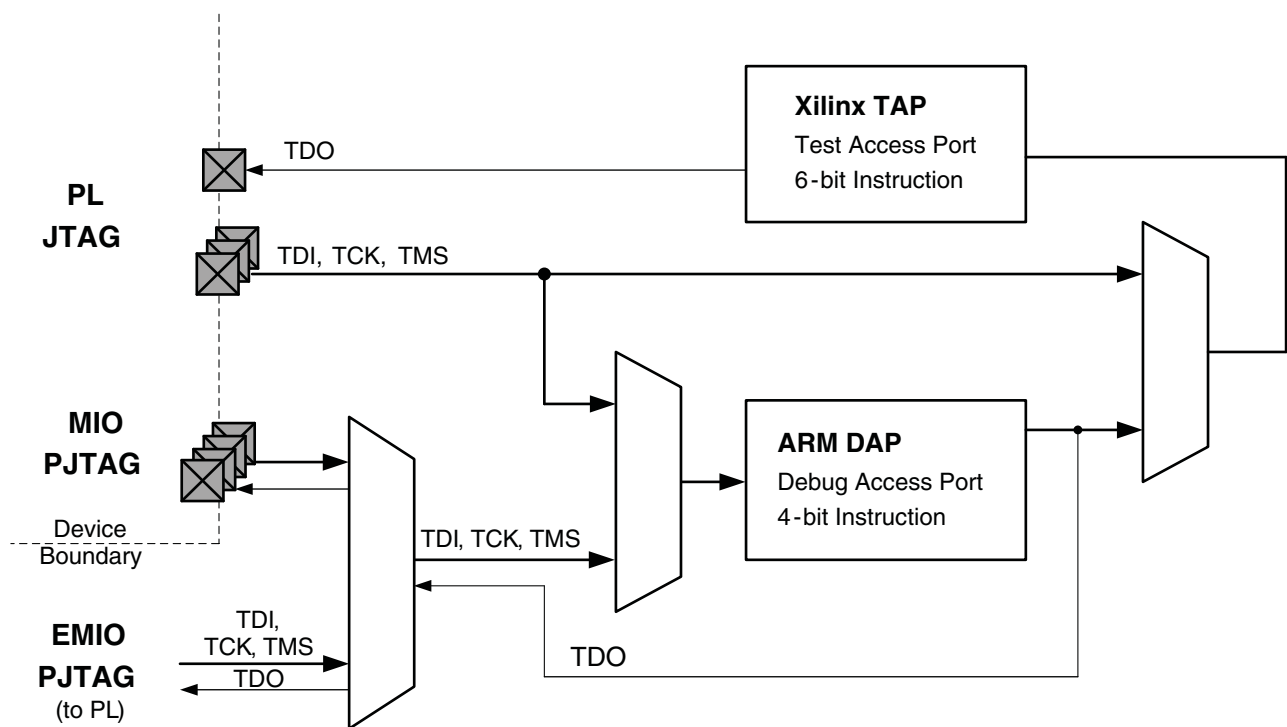
To debug ARM software and the PL design simultaneously with separate cables, the user must switch to independent JTAG mode. In this mode, JTAG cables only see the Xilinx TAP controller from the

dedicated PL PL_TDO/TMS/TCK/TDI pins. To debug ARM software, the user can route the ARM DAP signals (PJTAG) through the MIO or through EMIO and instantiate a soft core in the PL.

It is important to note that both the PS and PL must be powered on to use JTAG debug. Due to security reasons, the JTAG chain is protected with triple redundancy gating logic to prevent accidental debug enablement under the security environment due to a single event upset (SEU). Zynq-7000 EPP devices also provides JTAG disable lock-down to prevent debug enablement due to software errors.

The Zynq-7000 EPP provides for permanently disabling JTAG, by using one eFuse bit to record. Care should be used when selecting this option because the eFuse JTAG disable is not reversible.

Figure 27-2 shows the debug trace architecture. The user can enable debug trace source, PTM, ITM, and FTM using either the JTAG/DAP interface or software through the debug APB bus.



UG585_c27_01_042512

Figure 27-1: JTAG System Block Diagram

This section focuses on the trace port interface unit, which is one of the trace sink modules used to dump real-time trace to an external trace capture module. Both TPIU and ETB receive the exact the same copies of aggregated trace from multiple trace sources.

Although ETB is able to support high trace bandwidth, the 4 KB limit only allows capturing the trace in a small time window. To monitor trace information over a longer period of time, the user must enable the TPIU to dump either through MIO or EMIO so the trace is captured by external trace capture equipment such as an HP logic analyzer, Lauterbach Trace32, ARM DStream, etc.

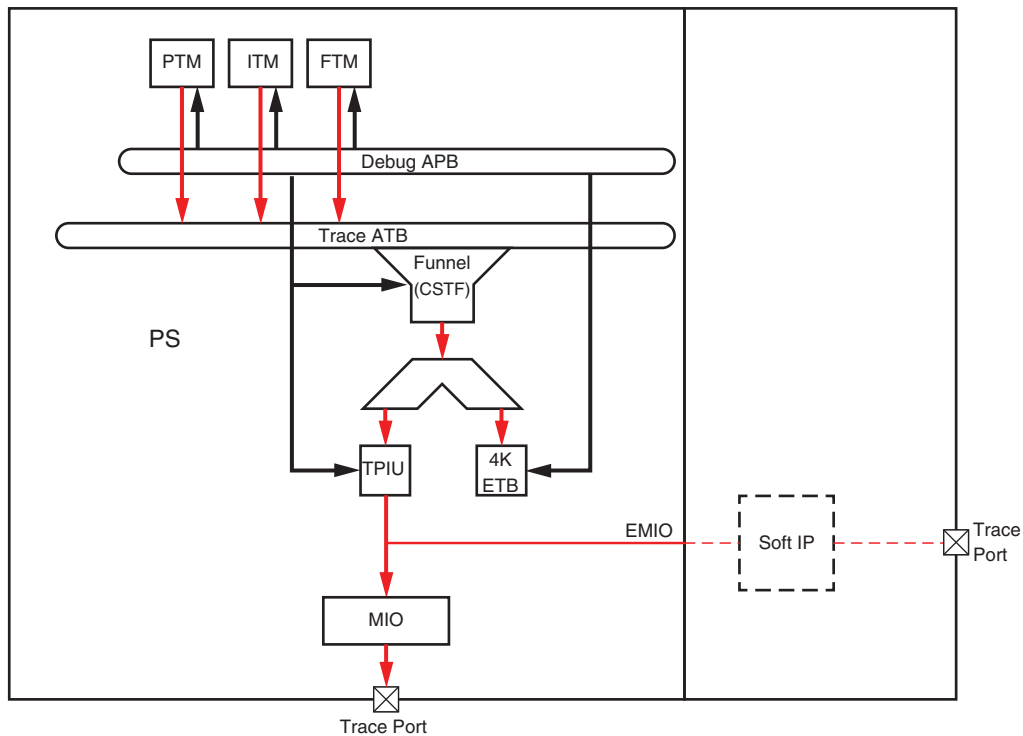


Figure 27-2: Debug Trace Port

27.1.2 Features

Key features of the JTAG debug interface are:

- JTAG 1149.1 boundary scan support
- Two 1149.1 compliance TAP controllers: One JTAG TAP controller and one ARM DAP
- Single unique IDCODE for each Zynq-7000 family
- IEEE 1532 programming in-system-configurable (ISC) devices support
 - eFuse programming
 - BBRAM programming
 - XADC access
- On-board flash programming
- Xilinx chipscope debug support
- ARM CoreSight debug center control using ARM DAP
- Direct system address space access through DAP-AP port
- External trace capture using MIO in PS, or EMIO in PL

27.2 Functional Description

Figure 27-3 shows the ARM DAP and JTAG TAP controllers connected in daisy-chain order with the ARM DAP at the front of chain. The two JTAG controllers belong to two different power domains. The ARM DAP is in the PS power domain while the TAP is in the PL power domain. JTAG I/O pads are located in the PL power domain to take advantage of existing JTAG I/O pads in the PL. Although the PS supports PL power down mode, both power domains must be powered on to support all JTAG related features.

The ARM DAP controller has a 4-bit IR length and the TAP controller has a 6-bit IR length. The two TAP and DAP controllers operate completely independently. The user can access the TAP and ARM DAP controller simultaneously in independent mode. Due to security reasons, the ARM DAP controller is bypassed when the PS is out of reset. The Xilinx TAP controller within the PL can be disabled through the eFuse or the control register within the PL configuration logic.

All debug components within the PS are under direct control of the debug tools, such as ARM RVDS or Xilinx XDK, through ARM DAP. All debug components (including DAP) within the PS are designed and integrated following ARM CoreSight architecture. Although there is no CoreSight component within the PL, FTM components within the PS allows a PL trace to be dumped into the ETB. CTI/CTM supports cross triggering between the PS and PL.

As shown in Figure 27-3, all PS debug components are tied to the debug APB bus with the DAP as the only bus master. External debug tools connected to the ARM DAP through JTAG uses the debug APB bus to configure all debug components including CPU, CTI/CTM, PTM, ITM, and FTM. Debug APB is also used to extract trace data from the ETB. It is a complicated process to configure all of the debug components properly to support user debug needs. Fortunately, most of the work is automatically handled by the debug tools. However, understanding Zynq debug architecture is necessary to better utilize the full system debug capability.

Other than debug control, the ARM DAP also acts as master device within the system interconnect. In previous debug systems, it was required to halt the CPU in order to probe system address space. This new arrangement allows the user to directly access system address space without halting the CPU. (See Chapter 28, System Test and Debug for more information).

The PL Xilinx tap controller serves four key purposes: boundary scan test, eFuse programming, BBRAM programming, and PL debug chipscope.

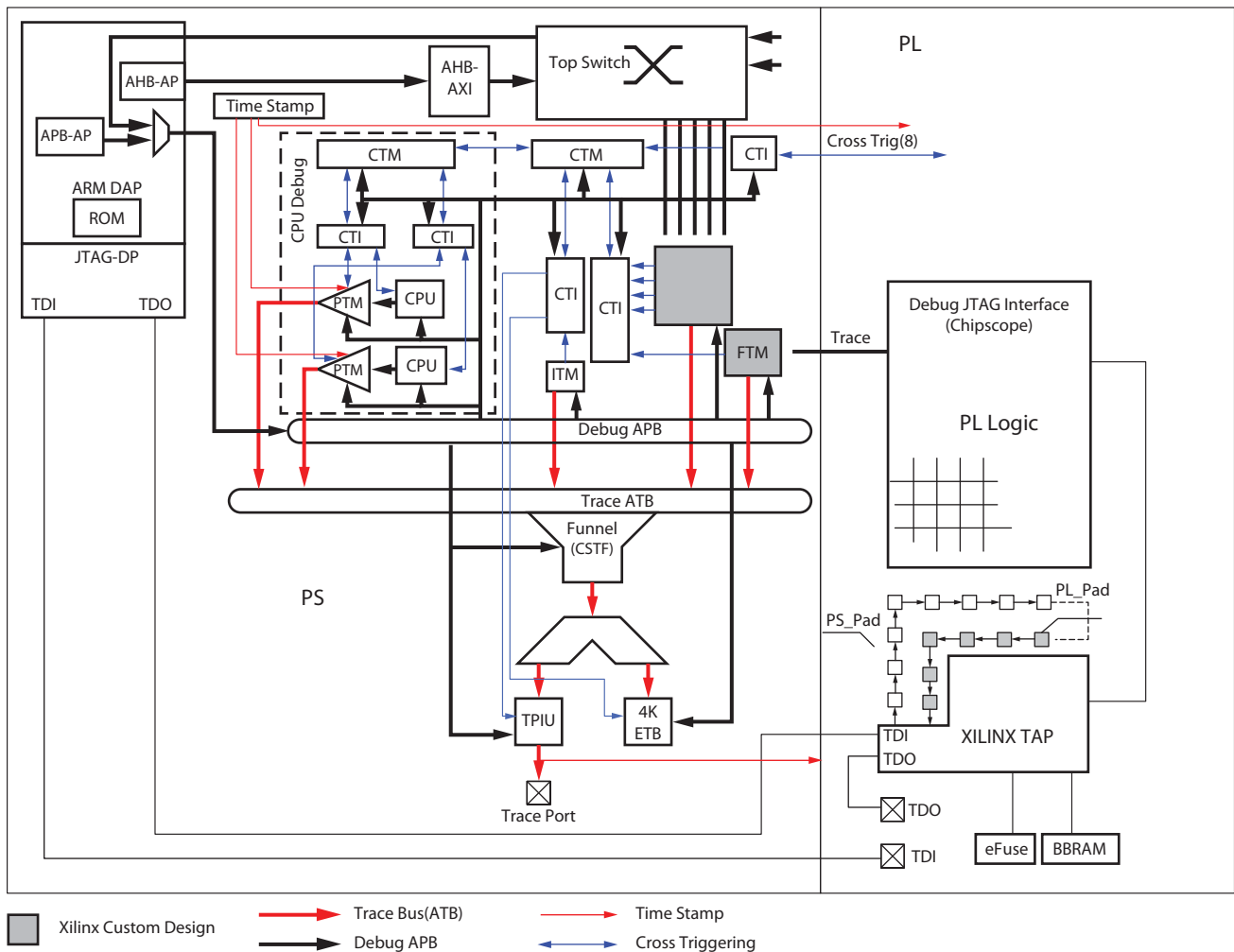


Figure 27-3: Debug System Architecture

The TPIU provides the mechanism to capture trace over long periods of time. There is no internal time limit to how long a trace can be dumped so the only practical limit is the Zynq-7000 bandwidth. If doing a trace dump using PS I/O through MIO, the maximum trace bandwidth depends on how many MIO trace I/Os could be allocated. Another alternative is trace dump through EMIO. PL soft logic connects the EMIO trace signal to the PL SelectIO. There are other potential innovative ways to handle EMIO trace.

For example, users could loopback EMIO trace data back to the PS and store it in DDR memory or export trace via gigabit Ethernet to enable remote debug or monitor. In typical debug flow, the user enables minimum trace source dumping capability to fit trace data into allocated TPIU throughput. After a small time window when debug occurs as determined through trace monitoring, the user could enable full trace dumping capability if required, and store short periods of data into the ETB for the next level of debug. Other than debugging, trace port also brings significant value for software profiling. Soft profiling helps the user to identify those software routines that consume the

most CPU power. Based on that, the user could decide to either perform software optimization or offload the process to the PL.

27.3 I/O Signals

Table 27-1 shows the PJTAG signals to access the ARM DAP and Xilinx TAP controllers. In cascaded JTAG mode, only PJTAG signals are meaningful to users. The user could access either the ARM DAP or Xilinx TAP through those signals. In JTAG independent mode, the user can only access Xilinx JTAG through the PJTAG signals as shown in Figure 27-1. To access the ARM DAP controller, the user must use PL Soft IP to bring EMIO signals to the SelectIO. The user must decide which SelectIO pins to use for the ARM DAP debug signals. The Zynq-7000 EPP does not support the SWD mode of the ARM DAP controller.

The MIO pins and any restrictions based on device version are shown in the MIO table in section 2.4.4 MIO-at-a-Glance Table.

Table 27-1: PJTAG Signals

Signal Name	via EMIO		PL JTAG via MIO	
	Signal	I/O	Pin	I/O
TCK	EMIOPJTAGTCK	I	MIO 12, 24, 36 or 48	I
TMS	EMIOPJTAGTMS	I	MIO 13, 25, 37 or 49	I
TDI	EMIOPJTAGTDI	I	MIO 10, 22, 34 or 46	I
TDO	EMIOPJTAGTDO	O	MIO 11, 23, 35 or 47	O
TDO 3-state	EMIOPJTAGTDTN	O		

Table 27-2 shows the ARM DAP/Xilinx TPIU signal routing.

Table 27-2: TPIU Signals

Signal Name	EMIO		MIO	
	Signal	I/O	Pin	I/O
TRACE_CLK	EMIOTRACECLK	I	MIO12 or MIO24	O
TRACE_CTL	EMIOTRACECTL	O	MIO 13 or MIO25	O
TRACE_DATA [1:0] TRACE_DATA [3:2] TRACE_DATA [7:4] TRACE_DATA [15:8]	EMIOTRACEDATA	O	MIO[15:14] or [27:26] MIO[23:22] MIO[19:16] MIO[9:2]	O

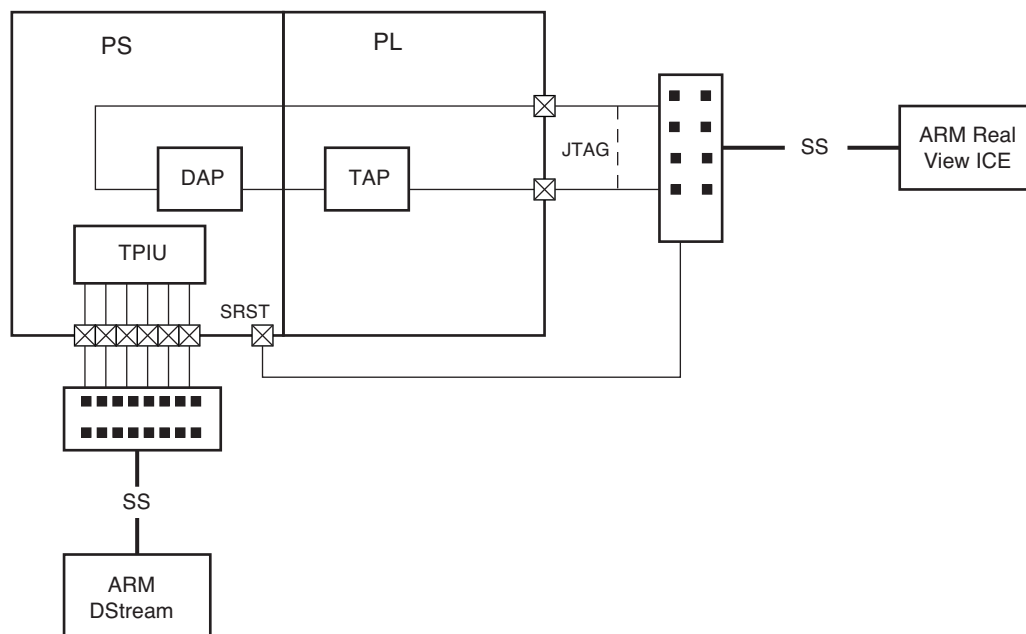
Notes:

With EMIO, the boundary for the I/O reference is the PS/PL boundary. With PJTAG, the boundary is the device package pins.

27.4 Programming Model

27.4.1 User Case I: Software Debug with Trace Port Enabled

This is the normal debug case for most applications. Figure 27-4 shows ARM tool chain solution. It is also possible to replace ARM Real View ICE with a Xilinx or Lauterbach debug tool. In this case, there is no PL programming required and the user is able to start on software debug as soon as chip power is on. In this use case, the DAP is active to support software debug needs but the TAP is put into bypass mode. Trace port is also enabled through the MIO pin. Although bandwidth might be limited due to limited MIO availability in some cases, the user could enable a trace dump without depending on PL configuration in this configuration. The major challenge for most users is to allocate MIO pins for the Trace port.



UG585_c27_04_031812

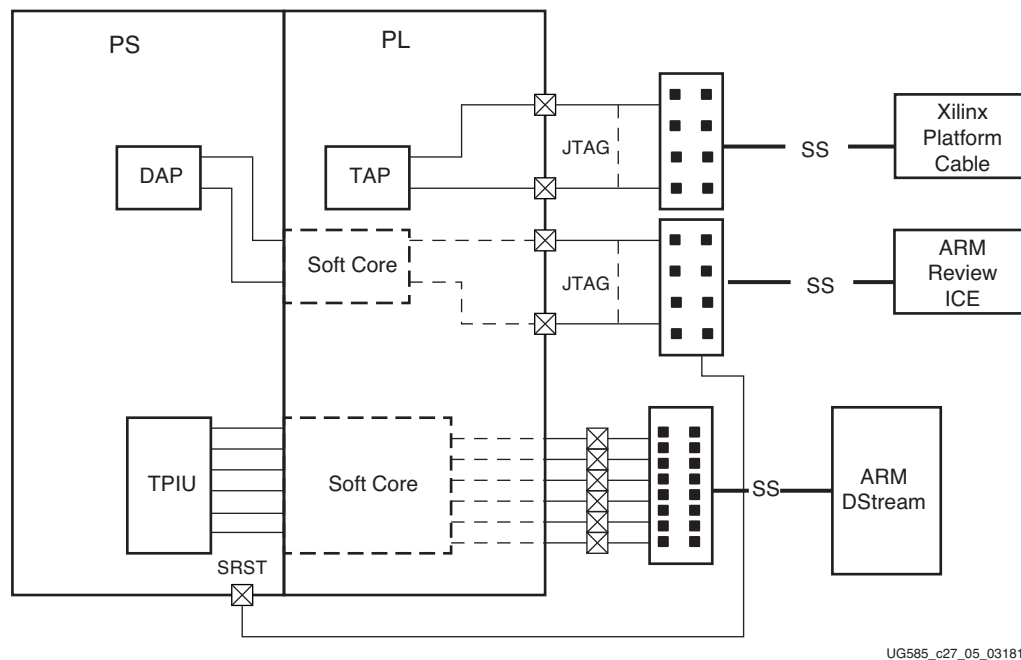
Figure 27-4: User Case I

27.4.2 User Case II: PS and PL Debug with Trace Port Enabled

The second use case shows how to enable PS software and PL hardware at the same time with two separate debug tools. The tool connected to the Xilinx TAP is typically a Xilinx debug tool. The tools connected to the PS DAP could be Xilinx or any third-party debug tools from ARM or Lauterbach.

To support this mode, PL configuration is required to bring the DAP JTAG signals to the PL SelectIOs. Figure 27-5 shows trace port access through the PL SelectIO, but the user could use the MIO trace port as in the previous use case if there is way to manage to fit the trace port into the limited MIO pin multiplexing. Trace port access through SelectIO could support up to 32-bit trace data and gives users enough trace port throughput to address most debug need. As with the JTAG DAP access, the

PL must be configured to route the trace signal from the EMIO at the PS/PL boundary to the PL SelectIO.



UG585_c27_05_031812

Figure 27-5: User Case II

27.5 ARM DAP Controller

The debug access port (DAP) is an implementation of an ARM debug interface standard comprising a number of components supplied in a single configuration. All of the supplied components fit into the various architectural components for debug ports (DPs) which are used to access the DAP from an external debugger, and access ports (APs) to access on-chip system resources.

With the JTAG-DP, IEEE 1149.1 scan chains are used to read or write register information. A pair of scan chain registers is used to access the main control and access registers within the debug port:

- DPACC used for debug port (DP) accesses
- APACC used for access port (AP) accesses

An APACC access might access a register of a debug component of the system to which the interface is connected. The scan chain model implemented by a JTAG-DP has the concepts of capturing the current value of APACC or DPACC, and of updating APACC or DPACC with a new value. An update might cause a read or write access to a DAP register that might then cause a read or write access to a debug register of a connected debug component.

Table 27-3 shows four Tap ARM DAP IR Instructions. All other IR Instructions are implemented as BYPASS.

Table 27-3: ARM DAP IR Instruction

IR Instruction	Binary Code[3:0]	DR Width	Description
ABORT	1 0 0 0	35	JTAG-DP abort register
DPACC	1 0 1 0	35	JTAG DP access register
APACC	1 0 1 1	35	JTAG-AP access Register
ARM_IDCODE	1 1 1 0	32	IDCODE for ARM DAP IP
BYPASS	1 1 1 1	1	

The ARM DAP is composed of one debug port (DP) and up to three access ports (APs). Among the three APs, Zynq-7000 devices only implement APB-AP as the bus master to access all debug components and AHB-AP to access system memory space directly. Table 27-4 lists all registers within the DP.

Table 27-4: DP Registers Summary

DP Register	Access	Description
CTRL/STAT	IR=DPACC/ADDRESS = 0x4	DP Control and Status register
SELECT	IR=DPACC/ADDRESS = 0x8	Its main purpose is to select the current access port and the active four-word register window in that access port

Table 27-5 shows AHB-AP and APB-AP registers in the DAP. For each AP, there is a unique set of registers associated with each AP port. Although the DAP allows JTAG-AP, Zynq devices do not support this feature.

Table 27-5: AP Registers Summary

AP Register	Access	Description
CSW	IR=APACC/ADDRESS = 0x0	Control and status word
TAR	IR=APACC/ADDRESS = 0x4	Transfer address, TAR
DRW	IR=APACC/ADDRESS = 0xC	Data read/write
BD0-3	IR=APACC/ADDRESS = 0x10 to 0x1C	Band data 0 to 3

27.6 Trace Port Interface Unit (TPIU)

Table 27-6 shows all registers within the TPIU.

Table 27-6: TPIU Registers Summary

TPIU Register	Offset	Description
SUPPORT_PORT_SIZE	0x0	32-bit register with each bit indicating whether a single port size is allowed
CURRENT_PORT_SIZE	0x4	Indicates current trace port size with only 1 of 32-bits that could be set
TRIG_MODE	0x100	Indicates trigger mode support
TRG_COUNT	0x104	8-bit register to enable delaying the indication of triggers to the external trace capture device
TRIG_MULT	0x108	Trigger counter multiplier
TEST_PATTERN	0x200–0x208	Configures a test pattern to generate a known bit sequence that could be capture by external capture device
FORMAT_SYNC	0x300–0x308	Control generation of stop, trigger, and flush events

27.7 Xilinx TAP Controller

The Xilinx TAP contains four mandatory dedicated pins as specified by the protocol and typical JTAG architecture. Table 27-7 shows Xilinx TAP IR commands. Refer to [UG470, 7 Series FPGAs Configuration User Guide](#) for more details about the IR commands.

Table 27-7: JTAG Commands

Boundary Scan Command	Binary Code[5:0]	Description
EXTEST	000000	Enables boundary-scan EXTEST operation
SAMPLE	000001	Enables boundary-scan SAMPLE operation
USER1	000010	Access user-defined register 1
USER2	000011	Access user-defined register 2
USER3	100010	Access user-defined register 3
USER4	100011	Access user-defined register 4
CFG_OUT	000100	Access the configuration bus for readback
CFG_IN	000101	Access the configuration bus for configuration
INTEST	000111	Enables boundary-scan INTEST operation
USERCODE	001000	Enables shifting out user code
IDCODE	001001	Enables shifting out of ID code

Table 27-7: JTAG Commands (Cont'd)

Boundary Scan Command	Binary Code[5:0]	Description
ISC_ENABLE	010000	Marks the beginning of ISC configuration; full shutdown is executed
ISC_PROGRAM	010001	Enables in-system programming
ISC_PROGRAM_SECURITY	010010	Change security status from secure to non-secure mode and vice versa
ISC_NOOP	010100	No operation
ISC_READ	101011	Used to read back BBR
ISC_DISABLE	010111	Completes ISC configuration. Startup sequence is executed
BYPASS	111111	Enables bypass

System Test and Debug

28.1 Introduction

The test and debug capability of Zynq-7000 EPP devices enables users to debug the PS and PL using intrusive and non-intrusive debug. Users can debug a complete system, including the PS and PL together. In addition to debugging software, users are also able to debug key hardware points in the PS and user-selected key hardware points in the PL.

The test and debug capability is based on the *ARM CoreSight v1.0 Architecture Specification* and consists mostly of ARM-supplied components, but also includes one Xilinx-supplied component. ARM CoreSight architecture defines four classes of CoreSight components: access and control, trace source, trace link, and trace sink.

Components of the access and control class provide a user interface to access the debug infrastructure through JTAG or memory-mapped locations. This class of components also coordinates the operation of separate CoreSight components with a trigger signal distribution network. Components of the trace source class capture debug information, like instruction addresses, bus transaction addresses, and generate trace packets. These trace packets are routed to components of the trace link class where trace packets can be combined or replicated. Components of the trace sink class receive trace packets and dump them into an on-chip trace buffer, or output them to chip pinouts (via the MIO) or to the PL (via the EMIO).

CoreSight components are connected together via three major types of buses/signals; programming, trigger, and trace. The programming bus is the path for the access and control class to convey programming information from the JTAG or from processors to other CoreSight components. The trigger signals are used by all classes of components to receive and send triggers from/to each other to coordinate their operation. The trace bus is the main pathway for trace packets to flow, connecting trace source, trace links, and trace sinks.

28.1.1 Features

The CoreSight components provide the following capabilities for the system-wide trace:

- Debug and trace visibility of whole systems with a single debugger connection
- Cross triggering support between SoC subsystems
- Multi-source trace in a single stream
- Higher data compression than previous solutions
- Standard programmer's models for standard tools support

- Automatic discovery of topology
- Open interfaces for third party soft cores
- Low pin count options

28.1.2 Notices

7z010 CLG225 Device

This device supports 32 MIO pins as shown in the MIO table in section 2.4.4 [MIO-at-a-Glance Table](#). The width of the TPIU in the 7x010 CLG225 device is restricted to 1, 2 or 4-bits via the MIO pins. All 32 data signals are available on the EMIO interface.

28.2 Functional Description

The block diagram for the CoreSight system is shown in [Figure 28-1](#).

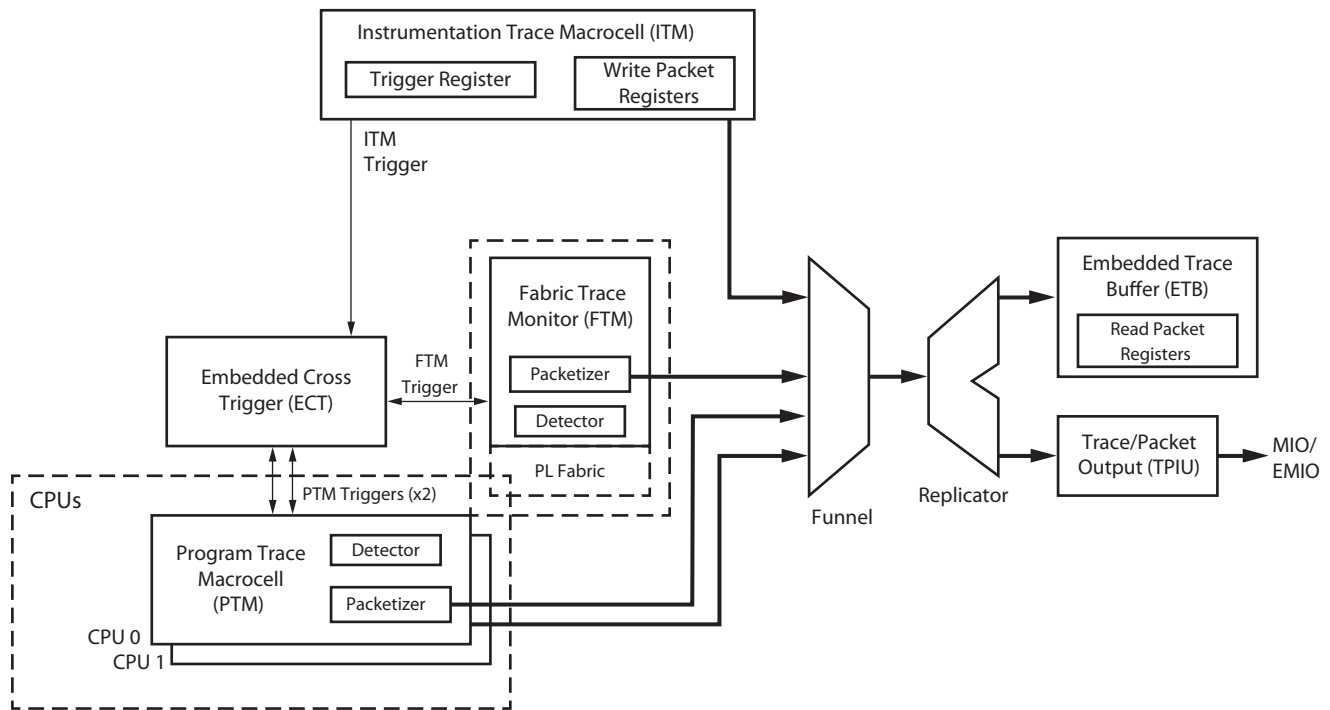


Figure 28-1: CoreSight System Block Diagram

Figure 28-1 shows the four classes of CoreSight components:

- Access and control: DAP, ECT

- Trace source: PTM, FTM, ITM
- Trace link: Funnel, Replicator
- Trace sink: ETB, TPIU

The components are connected by three types of buses/signals:

- Programming
- Trigger
- Trace

The CoreSight system interacts with:

- CPUs through PTM for debug and trace
- CPUs through ITM for trace
- PL through FTM for debug and trace
- CPUs through ETB for dumping trace
- EMIO/MIO through TPIU for dumping trace
- CPUs/JTAG through DAP for programming CoreSight components

28.2.1 Debug Access Port (DAP)

The DAP is the front-end for user access to the Zynq-7000s EPP system test and debug functionalities. It is a CoreSight component of the access and control class, and connects to other components using the programming bus. DAP provides the user, with two interfaces to access the CoreSight infrastructure:

- External: JTAG, from chip pinout
- Internal: APB slave, from the slave interconnect

A debugger can use JTAG to communicate with the CoreSight infrastructure, while software running on a CPU uses APB through memory-mapped addresses assigned to the CoreSight infrastructure. The DAP forwards access requests arriving via either interface to the requested CoreSight component.

In addition, the DAP also has another interface to access subsystems other than CoreSight, on the PS:

- Internal: AHB master, to the master interconnect

With AHB, the DAP can forward access requests from JTAG to other subsystems in the PS, subject to authentication requirements. For example, a debugger can query the value of a location in DDR or the value of a coprocessor register.

The DAP follows the access model described in *ARM Debug Interface v5 Architecture Specification* and *ARM Debug Interface v5.1 Architecture Supplement*, where JTAG indirectly accesses debug components and resources by way of registers in the DAP.

The DAP block is an ARM-supplied IP with the following configuration:

- JTAG is the only external interface on chip pinout. Serial wire interface (SW-DP) is not present.
- APB slave and AHB master are the two internal interfaces. JTAG at the DAP's internal side (JTAG-AP) is not present.
- Power-down is not supported.

28.2.2 Embedded Cross Trigger (ECT)

ECT is the cross-triggering mechanism. Through ECT, a CoreSight component can interact with other components by sending and receiving triggers. ECT is implemented with two components:

- CTM: Cross Trigger Matrix
- CTI: Cross Trigger Interface

One or more CTMs form an event broadcasting network with multiple channels. A CTI listens to one or more channels for an event, maps a received event into a trigger, and sends the trigger to one or more CoreSight components connected to the CTI. A CTI also combines and maps the triggers from the connected CoreSight components and broadcasts them as events on one or more channels. Both CTM and CTI are CoreSight components of the control and access class.

ECT is configured with:

- Four broadcast channels
- Five CTIs
- Power-down is not supported.

Table 28-1 lists the connections of trigger inputs and outputs of the CTIs.

Table 28-1: CTI Trigger Inputs and Outputs

CTI Trigger Port	Signal
CTI (connected to ETB, TPIU)	
Trigger input 2	ETB full
Trigger input 3	ETB acquisition complete
Trigger input 4	ITM trigger
Trigger output 0	ETB flush
Trigger output 1	ETB trigger
Trigger output 2	TPIU flush
Trigger output 3	TPIU trigger
CTI (connected to FTM)	
Trigger input 0	FTM trigger
Trigger input 1	FTM trigger
Trigger input 2	FTM trigger
Trigger input 3	FTM trigger
Trigger output 0	FTM trigger

Table 28-1: CTI Trigger Inputs and Outputs (Cont'd)

CTI Trigger Port	Signal
Trigger output 1	FTM trigger
Trigger output 2	FTM trigger
Trigger output 3	FTM trigger
CTI (connected to CPU0)	
Trigger input 0	CPU0 restarted
Trigger input 1	CPU0 PMU IRQ
Trigger input 2	PTM0 EXT
Trigger input 3	PTM0 EXT
Trigger input 4	CPU0 COMMTX
Trigger input 5	CPU0 COMMRX
Trigger input 6	PTM0 TRIGGER
Trigger output 0	CPU0 debug request
Trigger output 1	PTM0 EXT
Trigger output 2	PTM0 EXT
Trigger output 3	PTM0 EXT
Trigger output 4	PTM0 EXT
Trigger output 7	CPU0 restart request
CTI (connected to CPU1)	
Trigger input 0	CPU1 restarted
Trigger input 1	CPU1 PMU IRQ
Trigger input 2	PTM1 EXT
Trigger input 3	PTM1 EXT
Trigger input 4	CPU1 COMMTX
Trigger input 5	CPU1 COMMRX
Trigger input 6	PTM1 TRIGGER
Trigger output 0	CPU1 debug request
Trigger output 1	PTM1 EXT
Trigger output 2	PTM1 EXT
Trigger output 3	PTM1 EXT
Trigger output 4	PTM1 EXT
Trigger output 7	CPU1 restart request

28.2.3 Program Trace Macrocell (PTM)

The PTM is the block for tracing processor execution flow. It is based on the ARM program flow trace (PFT) architecture, and is a CoreSight component of the trace source class. The PTM generates information that trace tools use to reconstruct the execution of a program, by tracing only certain

points in program execution called waypoints. This reduces the amount of trace data. Timestamps are supported for correlating multiple trace streams and coarse grain code profiling.

The PTM provides some general resources, such as address/ID comparators, counter, sequencers, for setting up user-defined event triggering conditions. This enhances the base functions of tracing program execution.

The PTM block in Zynq-7000 EPP devices is the standard ARM-supplied IP, with the no custom configuration.

28.2.4 Instrumentation Trace Macrocell (ITM)

The ITM is the block for software to generate trace. It is a CoreSight component of the trace source class. Triggering and coarse-grained time stamping are also supported. The main uses include:

- printf style debugging
- Trace OS and application events
- Emit diagnostic system information

The ITM block in Zynq-7000 EPP devices is the standard ARM-supplied IP, with the no custom configuration.

28.2.5 Funnel

The funnel is the block for merging trace data from multiple sources into a single stream. It is a CoreSight component of the trace link class. Users select the sources to be merged and assign priorities to them.

The funnel in Zynq-7000 EPP devices is the standard ARM-supplied IP, with the no custom configuration.

[Table 28-2](#) lists the connections of the input ports of the Funnel.

Table 28-2: Funnel Input Port List

Port	Trace Source
0	PTM0
1	PTM1
2	FTM
3	ITM
4-7	unused

28.2.6 Embedded Trace Buffer (ETB)

The ETB is the on-chip storage of trace data. It is a CoreSight component of the trace sink class. The ETB provides real-time full-speed storing capability, but is limited in size. Triggering is supported for events such as buffer full and acquisition complete.

The ETB block in Zynq-7000 EPP devices is an ARM-supplied IP, with the following configuration:

- RAM size: 4 KB

28.2.7 Trace Packet Output (TPIU)

The TPIU is the block for outputting trace data to the PL or to chip pinout. It is a CoreSight component of the trace sink class. The TPIU provides unlimited trace data output, but is limited in bandwidth. Triggering and flushing are both supported.

The TPIU block in Zynq-7000 EPP devices is an ARM-supplied IP, with the following configuration:

- Max data width: 32

Table 28-3 shows the two operating modes of TPIU.

Table 28-3: Operating Modes of TPIU

SLCR[DBG_CLK_CTRL[6]]	Values	
	1	0
Active interface	EMIO	MIO
Clock source to operate the TPIU	EMIOTRACECLK	PS clock controller
Output clock present?	No	Yes
Clock edge(s) to sample trace data and control	Rising	Rising and falling
Supported data widths	1, 2, 4, 8, 16, 32	1, 2, 4, 8, 16
Application Note	Since PL supplies the clock to TPIU, PL can use the same clock to sample.	External device should delay the trace clock output by approximately half clock period, and use the delayed clock to sample.

Figure 28-2 shows the waveforms of TPIU I/O signals in the two modes. In EMIO mode, the PL supplies the trace clock signal to TPIU; PL can use the same clock to sample the trace data and control signals from PS. In MIO mode, all signals, including data, control, and clock, are aligned at the selected MIO pins; therefore, the external device should delay the trace clock output by approximately half the clock period to successfully sample the trace data and control signals.

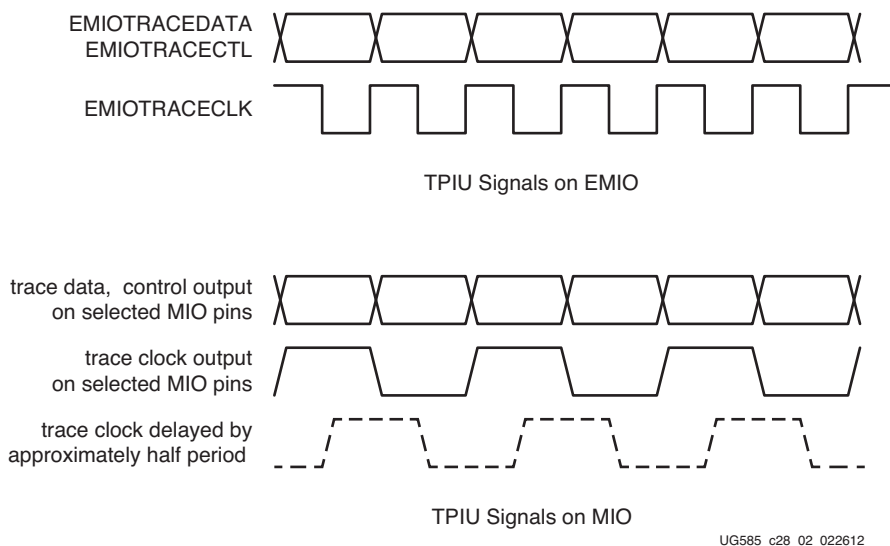


Figure 28-2: TPIU Operating Mode Waveforms

28.3 I/O Signals

Table 28-4 identifies the system test and debug I/O signals. The MIO pins and any restrictions based on device version are shown in the MIO table in section 2.4.4 MIO-at-a-Glance Table.

Table 28-4: Signals List

TPIU Signal	Default Input Value	MIO Pins			EMIO Signals	
		Number	I/O	Pin Name	Signal Name	I/O
Trace clock input	0	~	~	~	EMIOTRACECLK	I
Trace clock output	~	12 or 24	O	TRACE_CTL[0]	~	~
Trace control	~	13 or 25	O	TRACE_CTL[1]	EMIOTRACECTL	O
Trace data	~	1-bit	14 or 26	O	TRACE_DATA[15:0]	EMIOTRACEDATA[31:0]
		2-bit	15, 14 or 27, 26			
		4-bit	11, 10, 15, 14 or 23, 22, 27, 26			
		8-bit	19-16, 11, 10, 15, 14			
		16-bit	9-2,19-16,11,10,15,14			

28.4 Register Overview

28.4.1 Memory Map

Per the CoreSight specification, each CoreSight component has 4 kB address space. [Table 28-5](#) lists the base address of each CoreSight component.

Table 28-5: Memory Map

Component	Base Address
DAP ROM	0xF880_0000
ETB	0xF880_1000
CTI (connected to ETB, TPIU)	0xF880_2000
TPIU	0xF880_3000
Funnel	0xF880_4000
ITM	0xF880_5000
CTI (connected to FTM)	0xF880_9000
CTI (connected to AXIM)	0xF880_A000
FTM	0xF880_B000
Cortex-A9 ROM	0xF888_0000
CPU0 debug logic	0xF889_0000
CPU0 PMU	0xF889_1000
CPU1 debug logic	0xF889_2000
CPU1 PMU	0xF889_3000
CTI (connected to CPU0, PTM0)	0xF889_8000
CTI (connected to CPU1, PTM1)	0xF889_9000
PTM0 (for CPU0)	0xF889_C000
PTM1 (for CPU1)	0xF889_D000

28.4.2 Functionality

Table 28-6 summarizes the registers in each CoreSight component.

Table 28-6: CoreSight Component Register Summary

Function	Name	Overview
DAP ROM		
Pointers	Entry0-9	Pointers to other CoreSight components
CoreSight management	Peripheral ID0-7 Component ID0-3	These registers provide identification information
ETB		
Control	CTL	Enable/disable capture
Status	STS	Status on pipeline, acquisition, trigger, full/empty
RAM depth	RDP	Depth of RAM in words
RAM read	RRD, RRP	Read pointer and data
RAM write	RWD, RWP	Write pointer and data
Trigger counter	TRG	Sets the number of words to be stored after a trigger event
Formatter and flush	FFCR, FFSR	Stop events, trigger mark, flush start, formatting control
CoreSight management	Peripheral ID Component ID Device ID, type Claim, lock, authentication Integration test	These registers provide: <ul style="list-style-type: none"> • Identification information • Authentication and access control • Integration test
CTI		
Control	CONTROL	Enable/disable CTI
Acknowledge	INTACK	Provide for SW to acknowledge TRIGOUT when no hardware acknowledge is supplied
Channel event generation	APPSET, APPCLR, APPPULSE	Raise/clear/pulse channel events, used with GATE register to create local events
Channel event gating	GATE	Prevent the channel events from propagating to other CTI's in the system
Forwarding control	INEN, OUTEN	Enable the forwarding of events between the trigger interface and the channel interface
Trigger/Channel interface status	TRIGINSTATUS, TRIGOUTSTATUS, CHINSTATUS, CHOUTSTATUS	Provide the current status of the trigger and channel interfaces
CoreSight management	Peripheral ID Component ID Device ID, type Claim, lock, authentication Integration test	These registers provide: <ul style="list-style-type: none"> • Identification information • Authentication and access control • Integration test

Table 28-6: CoreSight Component Register Summary (Cont'd)

Function	Name	Overview
TPIU		
Supported feature		Show maximum and current values of supported port size, test patterns, etc.
Trigger		Set Trigger counter, multiplier
Testing		Set test pattern, modes and repeat count
Format and finish		Control and status of formatter and flush
CoreSight management	Peripheral ID Component ID Device ID, type Claim, lock, authentication Integration test	These registers provide: <ul style="list-style-type: none"> • Identification information • Authentication and access control • Integration test
Funnel		
Control	Control, Priority	Enable slave ports, set hold time, and priority
CoreSight management	Peripheral ID Component ID Device ID, type Claim, lock, authentication Integration test	These registers provide: <ul style="list-style-type: none"> • Identification information • Authentication and access control • Integration test
ITM		
Control	CR, SCR	Enable ITM, configure features like timestamp, sync packets, sync count, etc.
Stimulus	SPR	Cause the write data to be inserted into the FIFO for packets
Trace	TER, TTR	Enable trace and trigger
CoreSight management	Peripheral ID Component ID Device ID, type Claim, lock, authentication Integration test	These registers provide: <ul style="list-style-type: none"> • Identification information • Authentication and access control • Integration test
FTM		
Cortex-A9 Integration ROM		
Pointers	Entry	Pointers to other CoreSight components for A9
CoreSight management	Peripheral ID Component ID	These registers provide identification information
CPU debug		
Control	DBGDSCCR	Controls cache behavior while the CPU is in debug state
Breakpoints	BVR BCR	Set breakpoint values, and control breakpoints. A breakpoint can be set on an Instruction Virtual Address (IVA) or/and a Context ID
Watchpoints	WVR WCR	Set watchpoint values, and control watchpoints. A watchpoint can be set on a Data Virtual Address (DVA) or with a Context ID

Table 28-6: CoreSight Component Register Summary (Cont'd)

Function	Name	Overview
CoreSight management	Peripheral ID Component ID Device ID, type Claim, lock, authentication Integration test	These registers provide: <ul style="list-style-type: none"> • Identification information • Authentication and access control • Integration test
CPU PMU		
Control	PMCR	Performance monitor control
Status	PMOVSr	Overflow flag status
Counter	PMCNTENSET PMCNTENCLR PMSELR PMCCNTR	Counter enable set/clear, software increment, cycle count
Event counters	PMXEVTYPER PMXVCNTR	Counters to gather statistics on the operation of the processor and memory system
User enable	PMUSERENR	User enable
Interrupt Enable	PMINTENSET PMINTENCLR	Interrupt enable set/clear
PTM		
Configuration	ETMCR, ETMCCR, ETMTRIGGER, ETMSR, ETMSCR	Main control registers, configuration, set trigger events, and status
Trace Enable control	ETMSSSCR, ETMTTEVR, ETMTECR1	Trace enable start/stop, Trace enable event, Trace enable control
Address comparators	ETMACVR, ETMACTR	Address comparator values, types
Counters	ETMCNTRLDVR, ETMCNTENR, ETMCNTVR	Counter reload values, enable events, reload events, current values
Sequencers	ETMSQMNEVR, ETMSQR	Sequencer state transition events, sequencer current state
External output event	ETMEXTOUTEVER	Set events that control the corresponding external output
Context ID comparators	ETMCIDCVR1, ETMCIDCMR	Context ID comparator value, mask Sync frequency, ID
General control	ETMSYNCFR, ETMIDR	Sync frequency, ID
Misc.	ETMCCER, ETMEXTINSELr, ETMTSEVR, ETMAUXCR, ETMTRACEIDR, ETMOSLSR	Configuration code, external input selection, timestamp, auxiliary control, CoreSight trace ID, OS lock, power-down
CoreSight management	Peripheral ID Component ID Device ID, type Claim, lock, authentication Integration test	These registers provide: <ul style="list-style-type: none"> • Identification information • Authentication and access control • Integration test

28.5 Programming Model

28.5.1 Authentication Requirements

ARM CoreSight infrastructure uses four signals to control authentication:

- DBGEN Invasive debug enable
- NIDEN Non-invasive debug enable
- SPIDEN Secure invasive debug enable
- SPNIDEN Secure non-invasive debug enable

Table 28-7 lists the Zynq-7000 EPP device authentication requirements for the CoreSight components.

Table 28-7: CoreSight Authentication Requirements by Component

Requirement	DBGEN	NIDEN	SPIDEN	SPNIDEN
DAP AHB master				
Non-secure access	1	x	0	x
Secure access	1	x	1	x
CTI (connected to ETB, TPIU) CTI (connected to FTM)				
Enable all trigger inputs	x	x	x	x
Enable all trigger outputs	x	x	x	x
CTI (connected to CPU0) CTI (connected to CPU1)				
Enable trigger input 0	x	1	x	x
Enable trigger input 1	x	1	x	x
Enable trigger input 2	x	1	x	x
Enable trigger input 3	x	1	x	x
Enable trigger input 4	x	1	x	x
Enable trigger input 5	x	1	x	x
Enable trigger input 6	x	1	x	x
Trigger output 0	1	x	x	x
Trigger output 1	x	x	x	x
Trigger output 2	x	x	x	x
Trigger output 3	x	x	x	x
Trigger output 4	x	x	x	x
Trigger output 6	1	x	x	x
Trigger output 7	x	x	x	x

Table 28-7: CoreSight Authentication Requirements by Component (Cont'd)

Requirement	DBGEN	NIDEN	SPIDEN	SPNIDEN
Funnel				
TPIU				
ETB				
(no authentication is required)	x	x	x	x
ITM				
Disable stimulus registers 0-15	0	0	0	0
Disable stimulus registers 16-31	x	x	0	0
CPU debug				
CPU PMU				
Non-secure invasive debug	1	x	x	x
Non-secure non-invasive debug	1	x	x	x
	0	1	x	x
Secure invasive debug	1	x	1	x
Secure non-invasive debug	1	x	1	x
	1	x	x	1
	x	1	1	x
	x	1	x	1
PTM				
Prohibited regions of trace	Use the following to determine: if (\sim NIDEN & \sim DBGEN) Prohibited = 1 else if (security level == non-secure) Prohibited = 0 else if ((privilege level == user) & (SUNIDEN)) Prohibited = 0 else if (SPIDEN SPNIDEN) Prohibited = 0 else Prohibited = 1			

On-Chip Memory (OCM)

29.1 Introduction

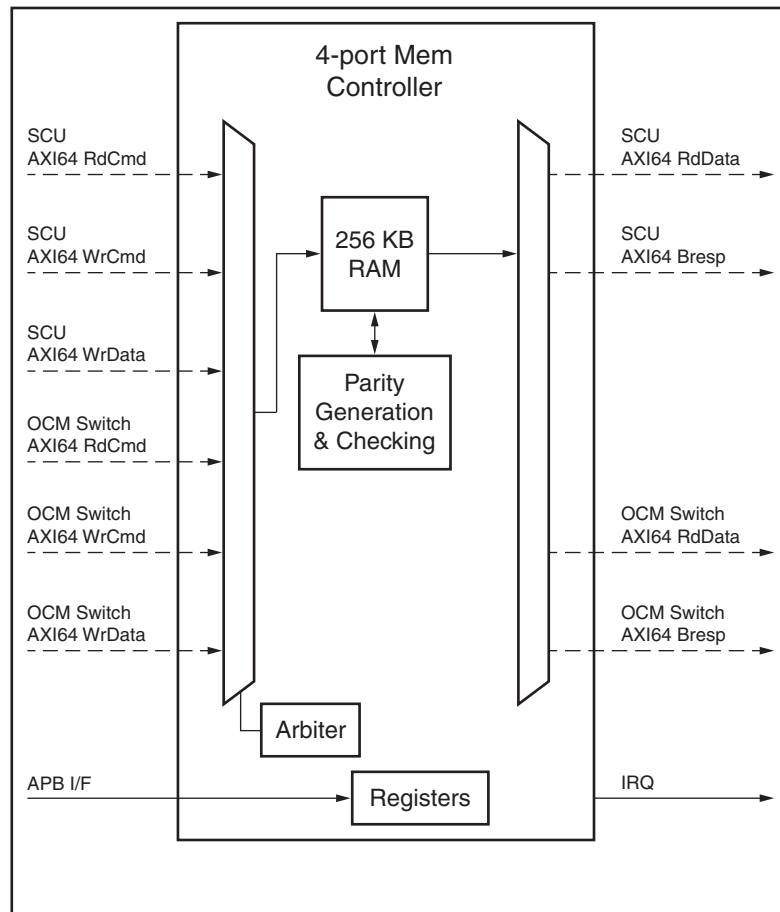
The on-chip memory (OCM) module contains 256 KB of RAM and 128K of ROM (BootROM). It supports two 64-bit AXI slave interface ports, one dedicated for CPU/ACP access via the APU snoop control unit (SCU), and the other shared by all other bus masters within the processing system (PS) and programmable logic (PL). The BootROM space is used exclusively by the boot process and is not visible to the user.

OCM supports high AXI read and write throughput for RAM access by implementing the RAM as a double-wide memory (128 bits). To take advantage of the high RAM access throughput, the user application must use even AXI burst sizes and 128-bit aligned addresses.

The TrustZone feature is supported at 4 KB memory granularity. The entire 256K RAM can be divided into 64 4 KB blocks, and assigned security attributes independently.

As shown in [Figure 29-1](#), there are 10 AXI channels associated with the OCM, 5 for the CPU/ACP (SCU) port and 5 for the other PS/PL masters (OCM switch port). Arbitration between the read and write channels of the SCU and OCM switch ports is performed within the OCM module. Parity generation and checking is performed on RAM accesses only. Other main interfaces are an interrupt signal (Irq) as well as a register access APB port.

29.1.1 Block Diagram



UG585_c29_01_042512

Figure 29-1: OCM Block Diagram

29.1.2 Features

Key features of the OCM include:

- On-chip 256 KB RAM
- On-chip 128 KB BootROM (not user visible)
- Two AXI 3.0, 64-bit slave interfaces
- Low latency path for CPU/ACP reads to OCM (CPU at 667 MHz – minimum 23 cycles)
- Round-robin pre-arbitration between read and write AXI channels on OCM-interconnect port (non-CPU port)
- Fixed priority arbitration between the CPU/ACP (via SCU) and OCM-interconnect AXI ports
- Supports full AXI 64-bit bandwidth of simultaneous read and write commands (with optimal alignment restrictions) on the OCM interconnect port
- Random access supported to RAM from AXI masters

- TrustZone support for on-chip RAM with 4 KB page granularity
- Flexible address mapping capability
- RAM byte-wise parity generation, checking, and interrupt support
- Support for the following non-AXI features on the CPU (SCU) port:
 - Zero line fill
 - Pre-fetch hint
 - Early BRESP
 - Speculative line pre-fetch

29.1.3 System Viewpoint

A system viewpoint of the OCM is illustrated in Figure 29-2.

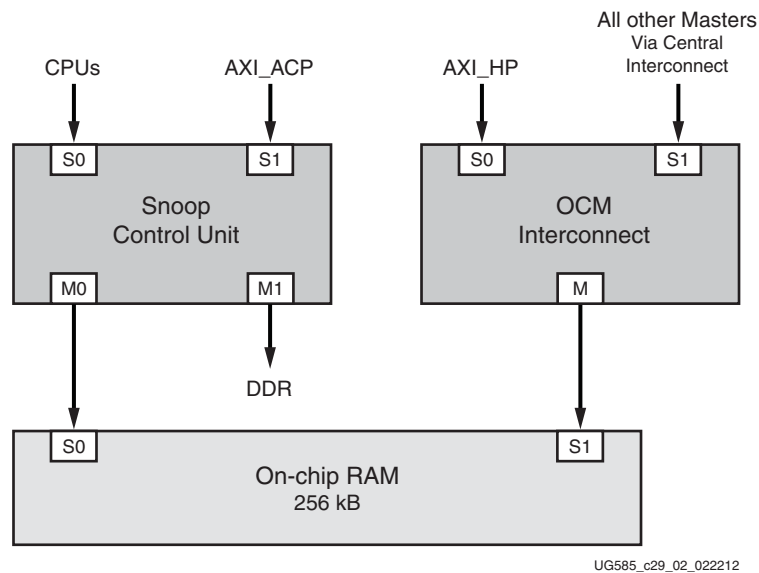


Figure 29-2: OCM System Viewpoint

29.2 Functional Description

29.2.1 Overview

The OCM module is mainly composed of a RAM memory block. The OCM module also contains arbitration, framing, parity, and interrupt logic in addition to the RAM array.

29.2.2 Optimal Transfer Alignment

The RAM is implemented as a single-ported, double-width (128-bit) module that can emulate a dual-ported memory under specific conditions. This emulation of dual-ported operation occurs automatically when 128-bit aligned, even burst multiples of AXI commands are used to access the 64-bit wide OCM AXI interfaces. Optimized bursts are theoretically able to achieve 100% throughput of the RAM. If bursts are not aligned to 128 bits or burst lengths are odd multiples of 64-bits, the control logic automatically realigns transfers inside the module — start and end addresses can be presented to the RAM as 64-bit operations instead of more optimal 128-bit operations.

Configuring OCM memory as *device memory* in the MMU or using narrow, non-modifiable accesses through the ACP port is not recommended. In this mode, pipelined 32-bit accesses are generated on the SCU port. This type of traffic pattern does not take advantage the double-width memory and effectively reduces OCM efficiency to 25%.

29.2.3 Clocking

The OCM module is clocked by the CPU_6x4x clock. However, the RAM array itself is an exception, and is clocked by CPU_2x, though its 128-bit width is double that of any of the incoming 64-bit wide AXI channels. The OCM switch feeding the OCM module is clocked by CPU_2x, and the SCU is clocked by CPU_6x4x.

29.2.4 Arbitration Scheme

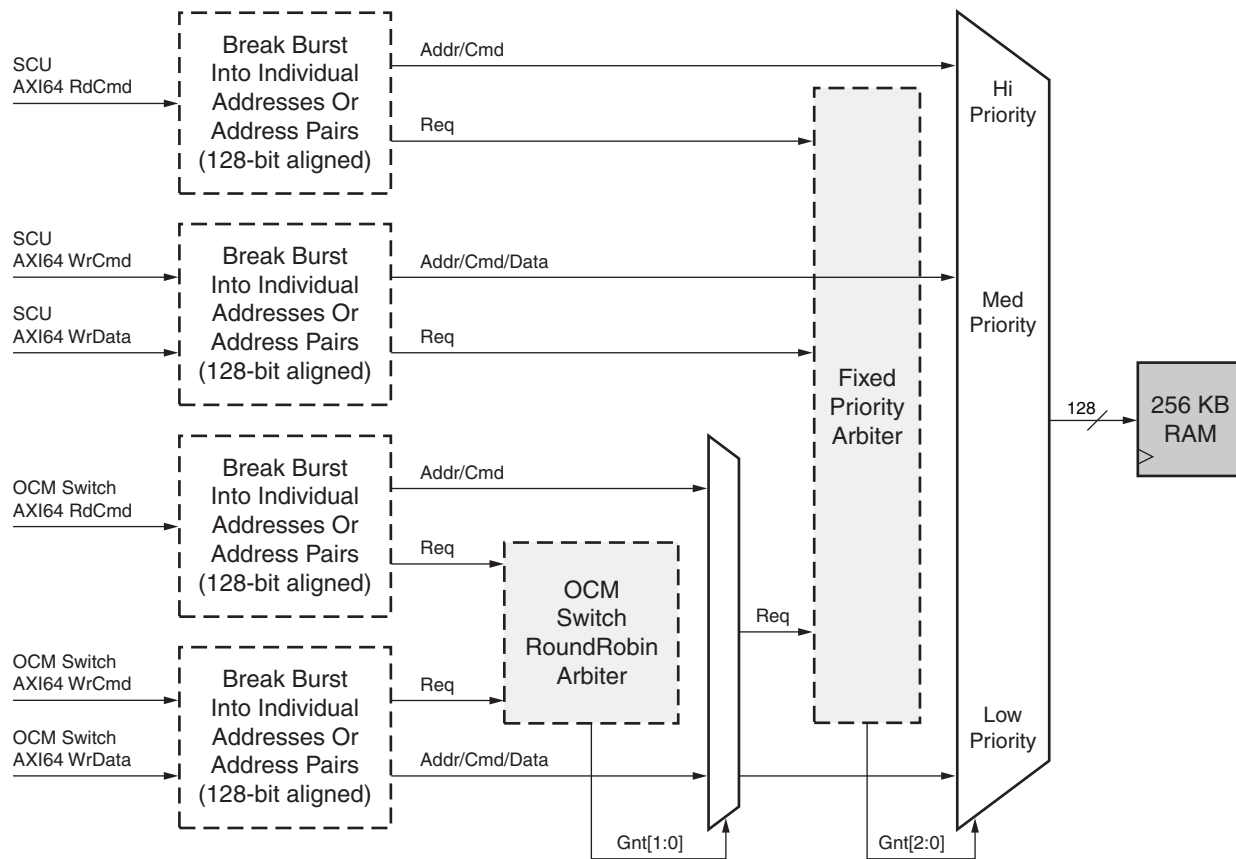
Apart from the CPUs and ACP, all other AXI bus masters are assumed to not have a strong latency requirement. Therefore, the OCM uses a fixed arbitration scheme (on a data beat basis) between the two AXI slave interfaces. The default order of decreasing priorities is:

1. SCU-Rd
2. SCU-Wr
3. OCM-Switch

Using the `ocm.OCM_CONTROL.ScuWrPriorityLo` register setting (see [Appendix B, Register Details](#)), the decreasing priority arbitration can be modified to:

1. SCU-Rd
2. OCM-Switch
3. SCU-Wr

Arbitration is implemented as shown in Figure 29-3.



UG585_c29_03_042512

Figure 29-3: Default (ScuWrPriorityLo=0) OCM Arbitration

There is an additional round-robin pre-arbitration process that selects between a read or write transaction on a per data-beat basis for the OCM-switch port traffic.

Note: Arbitration is performed on a transfer (data beat or clock cycle) basis, not on an AXI command basis. The in-coming AXI read and write commands are split into individual addresses (or 128-bit address pairs for aligned bursts) before arbitration.

Note: Each individual write address beat will not request access to the memory array until the write data associated with it is available inside the OCM module — this prevents the scenario of a write request being stalled due to write data not being available.

Starvation Scenarios

System constraints on the OCM are:

- The RAM array and OCM Switch port are clocked with CPU_2x which runs at one third or one half the CPU clock.
- The SCU (CPU/ACP) port is clocked at full the CPU clock rate.
- Each of the four incoming AXI data channels are 64-bits wide.

- The RAM array is 128-bits wide.
- The OCM switch port has separate read and write channels that can be simultaneously active.
- The SCU (CPU/ACP) port has separate read and write channels that can be simultaneously active.
- The SCU (CPU/ACP) port channels have a fixed arbitration priority higher than the OCM switch port by default.

The address range assigned to the OCM can be modified to exist in the first or last 256 KB of the address map, to flexibly handle the ARM low or high exception vector modes. In addition, the CPU and ACP AXI interfaces can have their lowest 1 MB address range accesses diverted to DDR, using the SCU address filtering feature. This section describes these features through a series of example address configurations.

Mapping Summary

(See [Appendix B, Register Details](#) for detailed register information.)

When addressing the OCM the following details should be considered:

- The 256 KB RAM array can be mapped to either a low address range (0x0000_0000 to 0x0003_FFFF) or a high address range (0xFFFFC_0000 to 0xFFFF_FFFF) in a granularity of four independent 64 KB sections via the 4-bit slcr.OCM_CFG[RAM_HI].
- The SCU address filtering (mpcore.SCU_CONTROL_REGISTER[Address_filtering_enable]) field is set by hardware on any form of reset and should not be disabled by the user. Address filtering on non-OCM addresses is necessary to correctly route transactions between the two downstream SCU ports. The address filtering range has a 1 MB granularity.
- The SCU address filtering feature is able to redirect accesses from its CPU and ACP masters targeting the range (0x0000_0000 to 0x000F_FFFF) which includes the OCM's low address range, to the PS DDR DRAM, independent of the RAM address settings.
- For each 64 KB section mapped to the high OCM address range via slcr.OCM_CFG[RAM_HI] which is not also part of the SCU address filtering range will be aliased for CPU and ACP masters at a range of 0x000C_0000-0x000F_FFFF.
- All other masters that do not pass through the SCU are always unable to access the lower 512 KB of DDR in the OCM's low address range (0x0000_0000 to 0x0007_FFFF).
- Accesses to addresses which the RAM array is not currently mapped to are given an error response.

Initial View

Upon entering user mode, the BootROM is no longer accessible, and the RAM space is split. Note that one 64 KB range resides at the high OCM address, and the other 192 KB resides at the lower address range. [Table 29-1](#) and [Table 29-2](#) identify the initial OCM/DDR address map and register settings, respectively.

Attempted accesses to reserved areas return all zeroes along with a SLVERR bus response.

Table 29-1: Initial OCM/DDR Address Map

Address Range (Hex)	Size	CPUs/ACP	Other Masters
0000_0000 - 0000_FFFF	64 KB	OCM	OCM
0001_0000 - 0001_FFFF	64 KB	OCM	OCM
0002_0000 - 0002_FFFF	64 KB	OCM	OCM
0003_0000 - 0003_FFFF	64 KB	Reserved	Reserved
0004_0000 - 0007_FFFF	256 KB	Reserved	Reserved
0008_0000 - 000B_FFFF	256 KB	Reserved	DDR
000C_0000 - 000C_FFFF	64 KB	Reserved	DDR
000D_0000 - 000D_FFFF	64 KB	Reserved	DDR
000E_0000 - 000E_FFFF	64 KB	Reserved	DDR
000F_0000 - 000F_FFFF	64 KB	OCM3 (alias)	DDR
0010_0000 - 3FFF_FFFF	1023 MB	DDR	DDR
FFFC_0000 - FFFC_FFFF	64 KB	Reserved	Reserved
FFFD_0000 - FFFD_FFFF	64 KB	Reserved	Reserved
FFFE_0000 - FFFE_FFFF	64 KB	Reserved	Reserved
FFFF_0000 - FFFF_FFFF	64 KB	OCM3	OCM3

Table 29-2: Initial Register Settings

Register	Value
slcr.OCM_CFG[RAM_HI]	1000
mpcore.SCU_CONTROL_REGISTER[Address_filtering_enable]	1
mpcore.Filtering_Start_Address_Register	0x0010_0000
mpcore.Filtering_End_Address_Register	0xFFE0_0000

OCM Relocation

For a contiguous RAM address range, RAM located at address 0x0000_0000 to 0x0002_FFFF can be relocated to base address 0xFFFF_C0000 by programming the SLCR registers.

Each bit of slcr.OCM_CFG[RAM_HI] corresponds to a 64 KB range, with the MSB corresponding to the highest address offset range. For more register programming details, refer to the SLCR information in the system level control registers section of [Appendix B, Register Details](#).

[Table 29-3](#) and [Table 29-4](#) identify an example OCM relocation address map and OCM relocation register settings, respectively.

Table 29-3: Example OCM Relocation Address Map

Address Range (Hex)	Size	CPUs/ACP	Other Masters
0000_0000 - 0000_FFFF	64 KB	Reserved	Reserved
0001_0000 - 0001_FFFF	64 KB	Reserved	Reserved

Table 29-3: Example OCM Relocation Address Map

Address Range (Hex)	Size	CPUs/ACP	Other Masters
0002_0000 - 0002_FFFF	64 KB	Reserved	Reserved
0003_0000 - 0003_FFFF	64 KB	Reserved	Reserved
0004_0000 - 0007_FFFF	256 KB	Reserved	Reserved
000C_0000 - 000C_FFFF	64 KB	OCM0 (alias)	DDR
000D_0000 - 000D_FFFF	64 KB	OCM1 (alias)	DDR
000E_0000 - 000E_FFFF	64 KB	OCM2 (alias)	DDR
000F_0000 - 000F_FFFF	64 KB	OCM3 (alias)	DDR
0010_0000 - 3FFF_FFFF	1023 MB	DDR	DDR
FFFC_0000 - FFFC_FFFF	64 KB	OCM0	OCM0
FFFD_0000 - FFFD_FFFF	64 KB	OCM1	OCM1
FFFE_0000 - FFFE_FFFF	64 KB	OCM2	OCM2
FFFF_0000 - FFFF_FFFF	64 KB	OCM3	OCM3

Table 29-4: Example OCM Relocation Register Settings

Register	Value
slcr.OCM_CFG[RAM_HI]	1111
mpcore.SCU_CONTROL_REGISTER[Address_filtering_enable]	1
mpcore.Filtering_Start_Address_Register	0x0010_0000
mpcore.Filtering_End_Address_Register	0xFFE0_0000

SCU Address Filtering

The view of the OCM as seen by the CPUs and ACP via the SCU port relative to other masters via the OCM switch is potentially different. The SCU uses its own dedicated address filtering mechanism to address slaves other than the OCM while the other bus masters in the system are routed via a fixed address decode scheme built into the system interconnects.

These other bus masters always see the OCM with accesses (from address 0x0000_000 to 0x0007_FFFF and address 0xFFFF_C000 to 0xFFFF_FFFF) going to OCM space. Depending on how the SLCR OCM registers are configured, these accesses either terminate at the RAM array or to a default reserved address, resulting in an AXI SLVERR error. These other masters potentially see gaps in the RAM address maps.

The CPU/ACP view, however, can be different using the SCU address filtering. For example, if the CPU wants DDR DRAM to be located at address 0x0000_0000, it can configure the address filtering and SLCR OCM registers so that the address map shown in Table 29-5 is seen. In Table 29-5, note that the CPU/ACP masters are able to address the entire DDR address range, while all other masters cannot address the lower 512 KB of DDR. Table 29-6 identifies example of OCM relocation register settings.

Table 29-5: Example SCU Address Filtering Address Map

Address Range (Hex)	Size	CPUs/ACP	Other Masters
0000_0000 - 0007_FFFF	512 KB	DDR	Reserved
0008_0000 - 000F_FFFF	512 KB	DDR	DDR
0010_0000 - 3FFF_FFFF	1023 MB	DDR	DDR
FFFC_0000 - FFFC_FFFF	64 KB	OCM0	OCM0
FFFD_0000 - FFFD_FFFF	64 KB	OCM1	OCM1
FFFE_0000 - FFFE_FFFF	64 KB	OCM2	OCM2
FFFF_0000 - FFFF_FFFF	64 KB	OCM3	OCM3

Table 29-6: Example OCM Relocation Register Settings

Register	Value
slcr.OCM_CFG[RAM_HI]	1111
mpcore.SCU_CONTROL_REGISTER[Address_filtering_enable]	1
mpcore.Filtering_Start_Address_Register	0x0000_0000
mpcore.Filtering_End_Address_Register	0xFFE0_0000

29.2.5 Interrupts

The OCM module is able to assert an interrupt signal to the APU under the following circumstances:

- Single-bit Parity Error
- Multiple-bit Parity Error
- Unsupported LOCK Request

All interrupts are enabled via the OCM.OCM_PARITY_CTRL register. Individual interrupt status is accessed via the OCM.OCM_IRQ_STS register, and cleared with a write of 1 to each bit location.

Parity on the RAM array is performed when the OCM.OCM_PARITY_CTRL[ParityCheckDis] is not asserted. When parity checking is enabled, a single- or multi-bit parity error sets the appropriate interrupt status, and triggers an external interrupt if the associated enable bit is set. The address offset of the first parity error is stored in the OCM.OCM_PARITY_ERRADDRESS register. For reads, a SLVERR response can also be issued to the requesting master for devices that are unable to or prefer not to handle interrupts.

29.3 Register Overview

A partial list of registers related to the OCM is listed in [Table 29-7](#). (See [Appendix B, Register Details](#) for the complete list.)

Table 29-7: On-Chip Memory Register Overview

Module	Register Name	Overview
OCM	OCM_PARITY_ERRADDRESS	Returns RAM parity error address
	OCM_PARITY_CTRL	Set interrupt enables, AXI read response error enable, parity enable, odd parity generation
	OCM_IRQ_STS	Read raw interrupt status, clear interrupts
	OCM_CONTROL	Change pre-arbitration priority
slcr	SLCR_LOCK	SLCR register write disable
	SLCR_UNLOCK	SLCR register write enable
	OCM_RST_CTRL	OCM subsystem reset
	TZ_OCM_RAM0/1	OCM TrustZone
	OCM_CFG	Configures RAM address mapping
mpcore	SCU_CONTROL_REGISTER	SCU address filtering enable
	Filtering_Start_Address_Register	SCU address filtering base address
	Filtering_End_Address_Register	SCU address filtering end address

29.4 Programming Model

29.4.1 Changing Address Mapping

A method for reorganizing the OCM address space and performing DDR remapping is as follows:

1. Complete all outstanding transactions by issuing data (DSB) and instruction (ISB) synchronization barrier commands.
2. Since the changing the address map may prevent the fetching of the nearby instructions, enable L1 instruction cache, and prefetch cache lines for the remainder of the function, typically with PLI instructions.
3. To facilitate prefetching, consider aligning the instructions to be prefetched to start at a cacheline boundary.
4. Ensure that the instruction prefetching has completed by issuing an ISB instruction.
5. Unlock the SLCR by writing the unlock key value to the slcr.SLCR_unlock register.
6. Modify the slcr.OCM_CFG register to change the address ranges that the RAM responds to.
7. Re-lock the SLCR by writing the lock key value to the slcr.SLCR_lock register, if desired.

8. Modify the `mpcore.Filtering_Start_Address_Register` to the desired start address of transactions that should be filtered away from the OCM for SCU masters. Typical settings are `0x0010_0000` (default, do not redirect lower 1 MB), and `0x0000_0000` (start redirect at lowest address to DDR RAM).
9. Modify the `mpcore.Filtering_End_Address_Register` to the desired end address of transactions that should be filtered away from the OCM for SCU masters. A typical setting is `0xFFE0_0000`.
10. Set `mpcore.SCU_CONTROL_REGISTER[Address_filtering_enable]` to enable address filtering.
11. Ensure that the access has completed to the SLCR by issuing a data memory barrier (DMB) instruction. This allows subsequent accesses to rely on the new address mapping.

29.4.2 AXI Responses

The OCM module produces the following AXI responses:

OKAY	Generated for exclusive access transactions, indicating exclusive access failure. OCM does not support exclusive access transactions.
DECERR	Generated for TrustZone (TZ) violations when accessing RAM/BootROM. A TZ violation occurs when a non-secure AXI access (<code>AxPROT[1] = 1</code>) is attempted to a secure 4K region of RAM/BootROM.
SLVERR	Generated for attempted access to reserved locations. SLVERR also generated for parity errors detected on RAM read access, if enabled.
OKAY	Generated for all other successful transactions.

Analog-to-Digital Converter Interface

30.1 Introduction

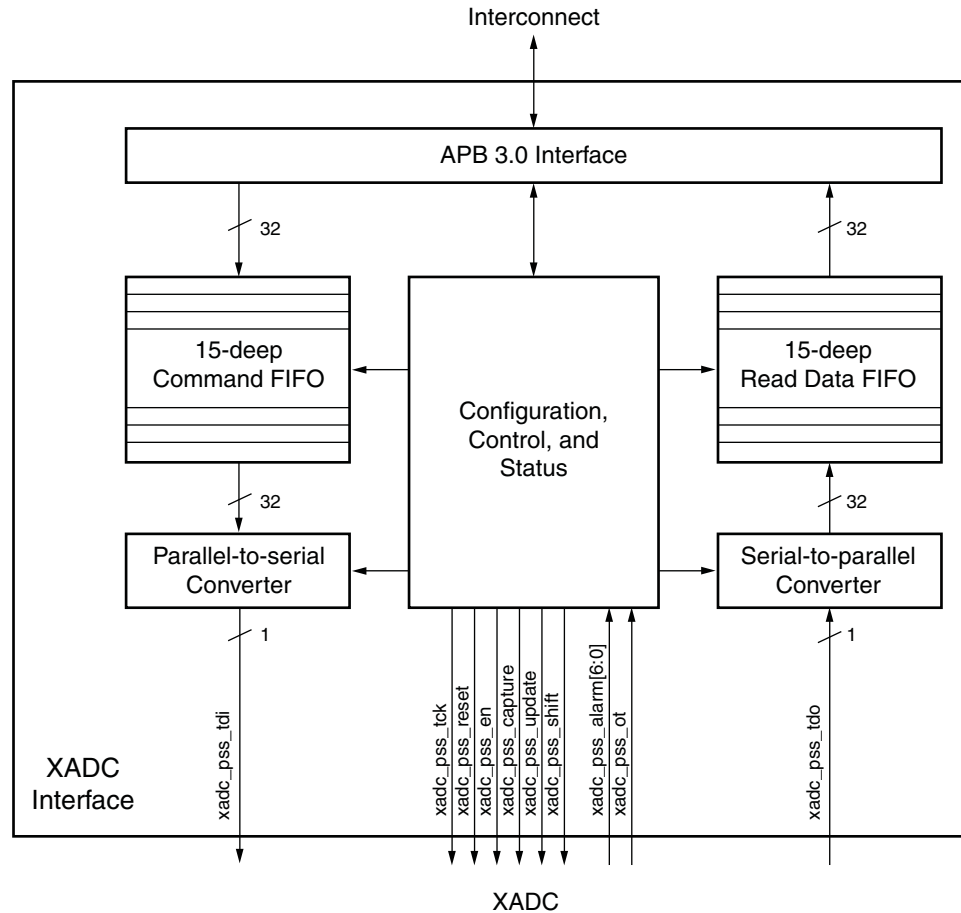
The analog-to-digital converter (XADC) interface, located within PS, allows the CPU and other hosts to access the XADC without the need for PL configuration. The XADC provides the full ADC function with sampling rates up to 1 MSPS and resolution up to 12-bits. Voltage and temperature sensing is also available. The PS communicates with the XADC module via the dedicated serial interface which serializes 32-bit commands from the PS for transmission to the PL. Concurrently, the interface block also de-serializes data from the PL into 32-bit quantities and presents them to the PS hosts via the APB interface. A command FIFO and a response FIFO are used to provide command and data buffering.

The XADC can also be accessed by the PS using the XADC AXI IP core. This core allows the XADC to be added as a soft peripheral attached to one of the AXI ports in the PS. See [UG480](#), *7 Series XADC User Guide* for details on operation of the XADC module.

Note: The interface must be enabled by setting a register bit to allow PS access to the XADC. When the PS controls the XADC, it is no longer accessible over the PL JTAG interface.

30.1.1 Block Diagram

Figure 30-1 shows the block diagram of the PS interface to the XADC.



UG585_c31_01_072512

Figure 30-1: XADC Interface Block Diagram

The dedicated interface between the PS and XADC is controlled by the PS. The PS sends 32-bit commands to the command FIFO which are in turn serialized and sent to the XADC DRP using a protocol based on the JTAG standard. Read commands result in data being sent serially to the PS interface from the XADC and stored in the read FIFO. The PS accesses the read data in 32-bit words. For more information, see the JTAG DRP interface section in [UG480, 7 Series XADC User Guide](#).

30.1.2 Features

The XADC interface provides the following functions and features:

- Read and write XADC registers
- 15-deep write command FIFO and 15-deep read FIFO (both 32-bit wide)
- Programmable FIFO-level interrupt
- Alarm interrupt

- Over-temperature interrupt
- Allows access to the XADC when the PL is not programmed (PL power is required)

The XADC interface operation are configured through a set of internal registers.

30.1.3 Notices

7z010 CLG225 Device

This device provides four ADC signal pairs (differential inputs). The hardware pin information is in the packaging and pinout document.

30.2 System Viewpoint

The XADC interfaces are connected to the PS's interconnect on one side and the PL on the other.

A host, such as the CPU, writes commands to be sent to the XADC into a memory-mapped location, and reads responses from the XADC via a separate memory-mapped location.

Alarms and over-temperature indications from the XADC are relayed to the PS via a maskable interrupt arrangement.

30.3 Functional Description

The XADC includes a dual 12-bit, 1 MSPS ADC and an on-chip voltage supply and thermal sensors. The ADCs can also access up to 17 external analog input channels. The PS accesses the XADC module through the XADC interface.

The XADC interface communicates with the XADC through a full duplex synchronous bit-serial link with dedicated control signals using a boundary scan protocol. The interface interacts with the rest of the PS via an APB 3.0 interface.

The main function of the XADC interface is to serialize commands before sending them to the XADC, and to perform serial-to-parallel conversion when serial data is received from the XADC. By default, after reset, the connection between the PS and XADC is disabled. Bit 31 of the Configuration register at offset 0x100 must be set to 1 to establish the link between the PS and PL. When the PS controls the XADC, it is no longer accessible over the PL JTAG interface.

The XADC supports the PS XADC interface (JTAG DRP) and direct access via the PL. Both the dedicated connection and the PL connection can be used simultaneously within the system. The XADC is capable of arbitrating between the PS XADC interface and a PL connection.

To minimize the effects of the slow serial transfer process on PS system throughput, the XADC interface provides for buffering of up to 15 32-bit commands to the XADC and 15 32-bit read data

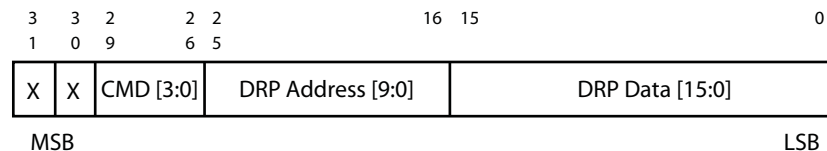
from the XADC. This allows a master to send up to 15 commands at a time and then return later to retrieve the data when it is ready. After a read command has been sent to the XADC, the corresponding read data will be available during the next shift period. Therefore, one dummy command is always needed to push out the last read data.

The status of the command and read FIFOs can be tracked in the XADC Interface Miscellaneous Status register (offset 0x10C) or setup to issue interrupts to the CPU using the XADC Interface Interrupt Status register (offset 0x104). For a full list of XADC commands, see [UG480, 7 Series XADC User Guide](#).

Note: Reading from an empty read data FIFO causes an APB slave error.

XADC Command Format

When active, the PS XADC interface replaces the PL JTAG DRP interface. All PS XADC interface commands are 32-bits wide. [Figure 30-2](#) shows the data format of the PS XADC interface commands. The first 16 LSBs of the XADCIF_CMD_FIFO contain the DRP register data. For both read and write operations, the address bits, XADCIF_CMD_FIFO[25:16], hold the DRP target register address. The command bits, XADCIF_CMD_FIFO[29:26], specify a read, write, or no operation (see [Table 30-1](#)). The full list of XADC DRP registers and instructions on how to configure them can be found in [UG480, 7 Series XADC User Guide](#).



UG585_c31_02_031812

Figure 30-2: XADC Interface DRP Command

Table 30-1: XADC Interface DRP Commands

CMD[3:0]				Operation
0	0	0	0	No operation
0	0	0	1	DRP read
0	0	1	0	DRP write
-	-	-	-	Not defined

Supply Sensor Alarms

The XADC tracks the minimum and maximum values recorded for the internal supply sensors since the last power-up or the last reset of the XADC control logic. The maximum and minimum values recorded are stored in the DRP Status registers. On power-up or after reset, all Minimum registers are set to FFFFh and all Maximum registers are set to 0000h. Each new measurement generated for an on-chip sensor is compared to the contents of its Maximum and Minimum register. If the measured value is greater than the contents of its Maximum register, the measured value is written to the Maximum register. Similarly, for the Minimum register, if the measured value is less than the contents of its Minimum register, the measured value is written to the Minimum register. This check is carried out each time a measurement result is written to the Status register.

The XADC can generate an alarm signal when an internal sensor measurement exceeds some user-defined thresholds. The alarm thresholds are stored in the XADC's Control registers and the alarms can be disabled in the XADC's Configuration register. When the measured value on the supply sensor is greater than the maximum thresholds or less than the minimum thresholds set in the Control registers, the output alarms go active. The alarms are reset when a subsequent measured value falls inside the threshold.

The XADC alarm signals are sent to the PS via the dedicated XADC interface. When an alarm goes active it triggers a maskable interrupt. The XADCIF_INT_STS register can be used to determine which alarm was activated. Writing a 1 to the active alarm bit in the XADCIF_INT_STS clears the interrupt flag. The real time value of alarm signal can be found on the XADCIF_MSTS register. Table 30-2 shows the available alarm signals

Table 30-2: XADC Alarm Signals

Alarm	Description
ALM[0]	XADC temperature sensor alarm
ALM[1]	XADC V_{CCINT} sensor alarm
ALM[2]	XADC V_{CCAUX} sensor alarm
ALM[3]	XADC V_{CCBRAM} sensor alarm
ALM[4]	XADC V_{CCPINT} sensor alarm (Processor V_{CCINT})
ALM[5]	XADC V_{CCPAUX} sensor alarm (Processor V_{CCPAUX})
ALM[6]	XADC V_{CCDDRO} sensor alarm (Processor DDR controller voltage)

Thermal Management

The on-chip temperature measurement is used for critical temperature warnings. The default *over temperature* threshold is 125°C. This threshold is used when the contents of the OT Upper Alarm register have not been configured. When the die temperature exceeds the threshold set in the XADC's Control register, the over-temperature alarm (OT) becomes active. The OT signal resets when the die temperature has fallen below set threshold. The OT alarm can also be used to automatically power down the PL upon activation. The OT alarm can be disabled by writing a 1 to the OT bit in the XADC's Configuration register.

The XADC OT alarm signal is sent to the PS via the dedicated XADC interface. When the OT alarm goes active it triggers a maskable interrupt. The OT bit of the XADCIF_INT_STS register needs to be cleared by writing a 1 to the bit location. The real time value of the OT alarm signal can be found on the XADCIF_MSTS register.

30.4 Register Overview

Table 30-3 shows an overview of XADC Interface registers.

Table 30-3: Register Overview

Address Offset	Mnemonic	Description	Type
0x100	XADCIF_CFG	XADC Interface Configuration register	R/W
0x104	XADCIF_INT_STS	XADC Interface Interrupt Status register	R + Clr on Wr
0x108	XADCIF_INT_MASK	XADC Interface Interrupt Mask register	R/W
0x10C	XADCIF_MSTS	XADC Interface Misc. Status register	RO
0x110	XADCIF_CMDFIFO	XADC Interface Command FIFO	WO
0x114	XADCIF_RDFIFO	XADC Interface Read Data FIFO	RO
0x118	XADCIF_MCTL	XADC Interface Misc. Control register	R/W

30.5 Programming Model

The following describes how to configuration and interact with the XADC interface.

- Set the configuration register (address offset 0x100)
 - Set the read and write FIFO threshold for triggering interrupts
 - Set the clock rate going to the XADC, the XADC clock is sourced from the PCAP clock
 - Set the clock edge for write launch and read capture
 - Set the minimum gap between successive commands
 - Set the enable bit to activate the link
- Write commands and read data using the FIFOs
- Monitor the FIFO status with interrupts or the status register
- The XADC-PS communication link can be reset using the reset bit on the Miscellaneous Control register.

PCI Express

31.1 Introduction

The Zynq-7030 and Zynq-7045 EPP devices include the Xilinx 7 series integrated block for PCI Express core which is a reliable, high-bandwidth, third-generation I/O solution.

The PCI Express solution for these Zynq EPP devices supports x1, x2, x4, and x8 lane Root Port and Endpoint configurations at both Gen1 (2.5 Gb/s) and Gen2 (5 Gb/s) speeds. The Root Port configuration can be used to build a Root Complex solution. These configurations are compatible with the PCI Express Base Specification, Rev 2.1.

The PCI Express module supports the AXI4-stream interface for the user interface at both 64-bit and 128-bit widths.

For more detailed information regarding the 7 Series FPGAs Integrated block for PCI Express core, refer to these documents on the Xilinx website:

- [DS821](#), *LogiCORE IP 7 Series FPGAs Integrated Block for PCI Express Data Sheet*
- [UG477](#), *LogiCORE IP 7 Series FPGAs Integrated Block for PCI Express User Guide*

31.2 Block Diagram

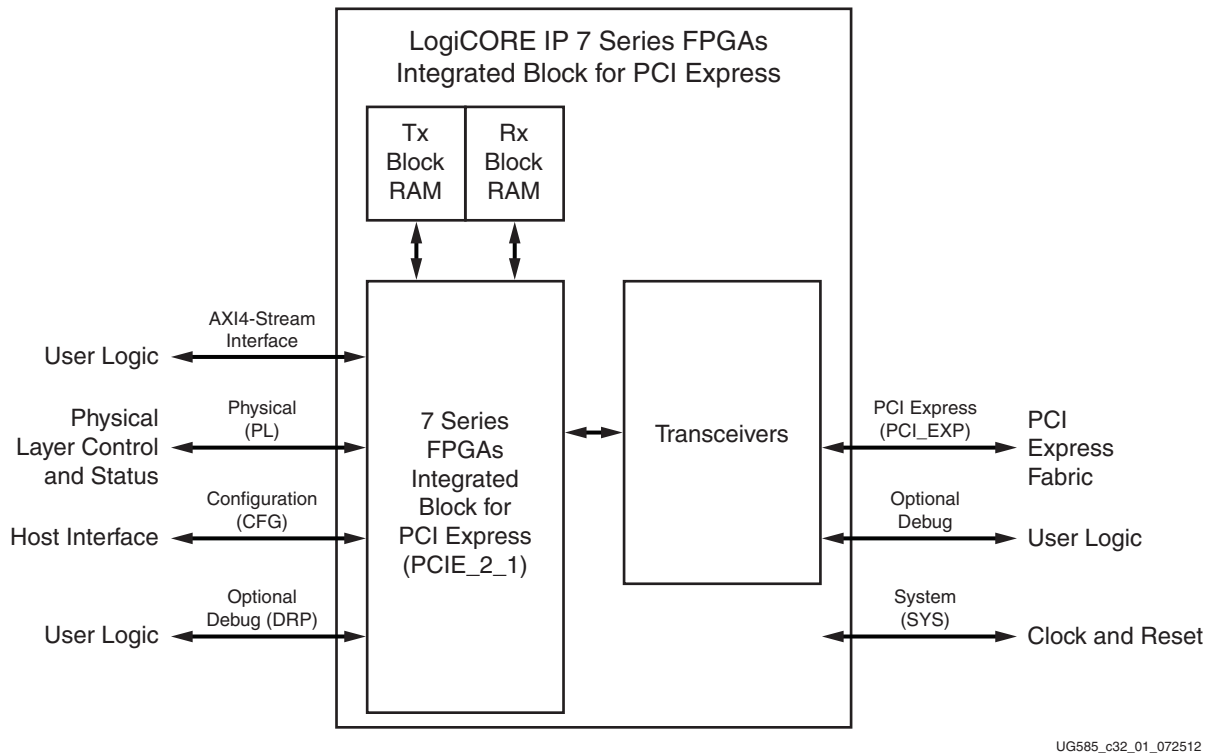


Figure 31-1: PCI Express Block Diagram

31.3 Features

The PCI Express core provides these key features:

- High-performance, highly flexible, scalable, and reliable, general-purpose I/O core
 - Compatible with the PCI Express Base Specification, rev. 2.1
 - Compatible with conventional PCI software model
- Incorporates Xilinx® Smart-IP™ technology to guarantee critical timing
- Uses GTXE2 transceivers for 7 Series FPGA families
 - 2.5 Gb/s and 5.0 Gb/s line speed
 - Supports 1-lane, 2-lane, 4-lane, and 8-lane operation
 - Elastic buffers and clock compensation
 - Automatic clock data recovery
- Supports Endpoint and Root Port configurations

- 8B/10B encode and decode
- Supports lane reversal and lane polarity inversion per PCI Express specification requirements
- Standardized user interface
 - Supports AXI4-stream interface
 - Easy-to-use packet-based protocol
 - Full-duplex communication
 - Back-to-back transactions enable greater link bandwidth utilization
 - Supports flow control of data and discontinuation of an in-process transaction in transmit direction
- Supports flow control of data in receive direction
- Compatible with PCI/PCI Express power management functions
- Supports a maximum transaction payload of up to 1024 bytes
- Supports multi-vector MSI for up to 32 vectors and MSI-X
- Up-configure capability enables application-driven bandwidth scalability
- Compatible with PCI Express transaction ordering rules

Device Secure Boot

32.1 Introduction

Zynq-7000 EPP devices support the ability to perform a secure boot to load encrypted PS images and PL bitstreams.

32.1.1 Block Diagram

[Figure 32-1](#) is a block diagram showing the different systems involved in a secure boot.

32.1.2 Features

Zynq-7000 EPP devices provide the following secure boot features:

- AES-256 cipher block chaining (CBC) decryption engine
 - Encryption key stored on-chip in either eFuse or battery-backed RAM (BBRAM)
- Keyed-hashed message authentication code (HMAC)
 - SHA-256 (secure hash algorithm) authentication engine
 - FIPS180-2

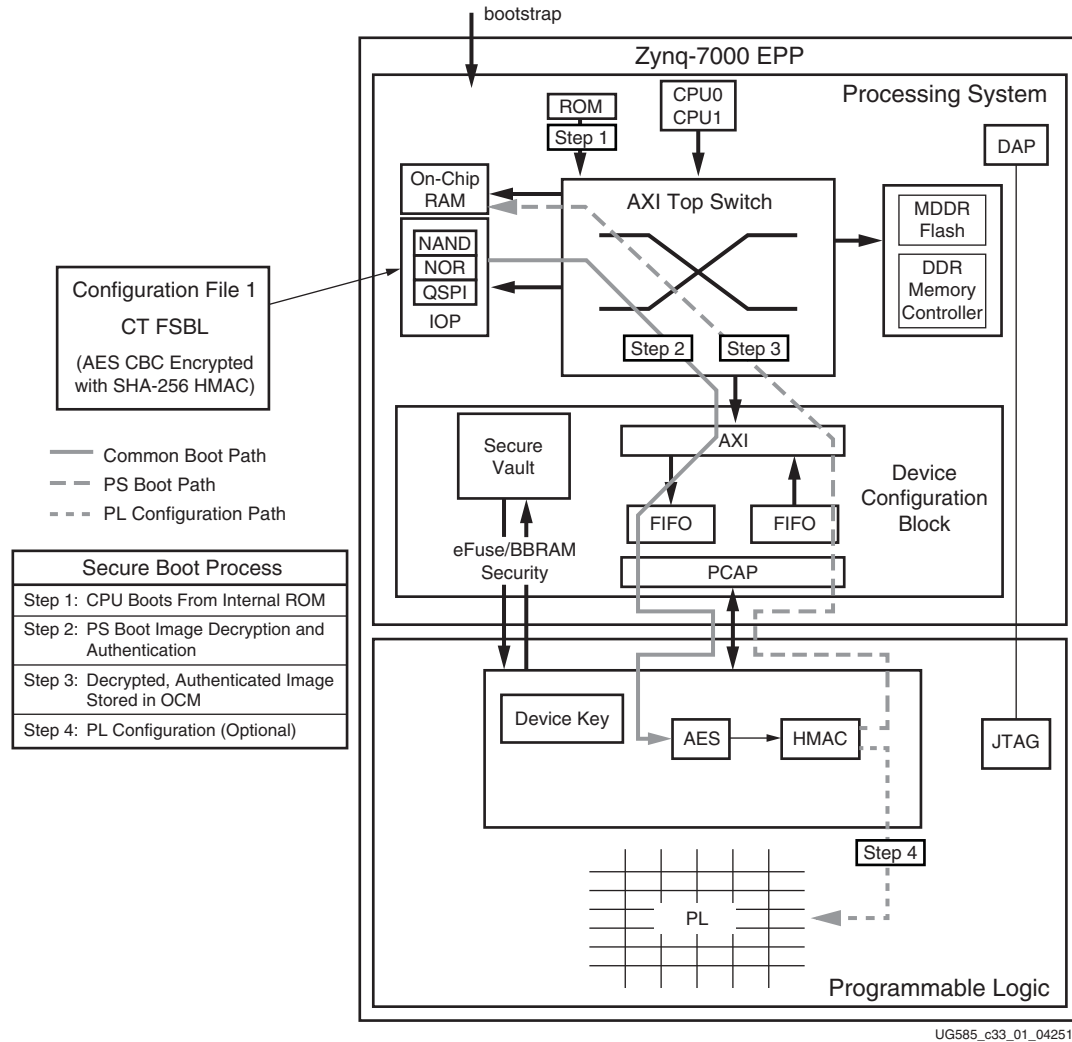


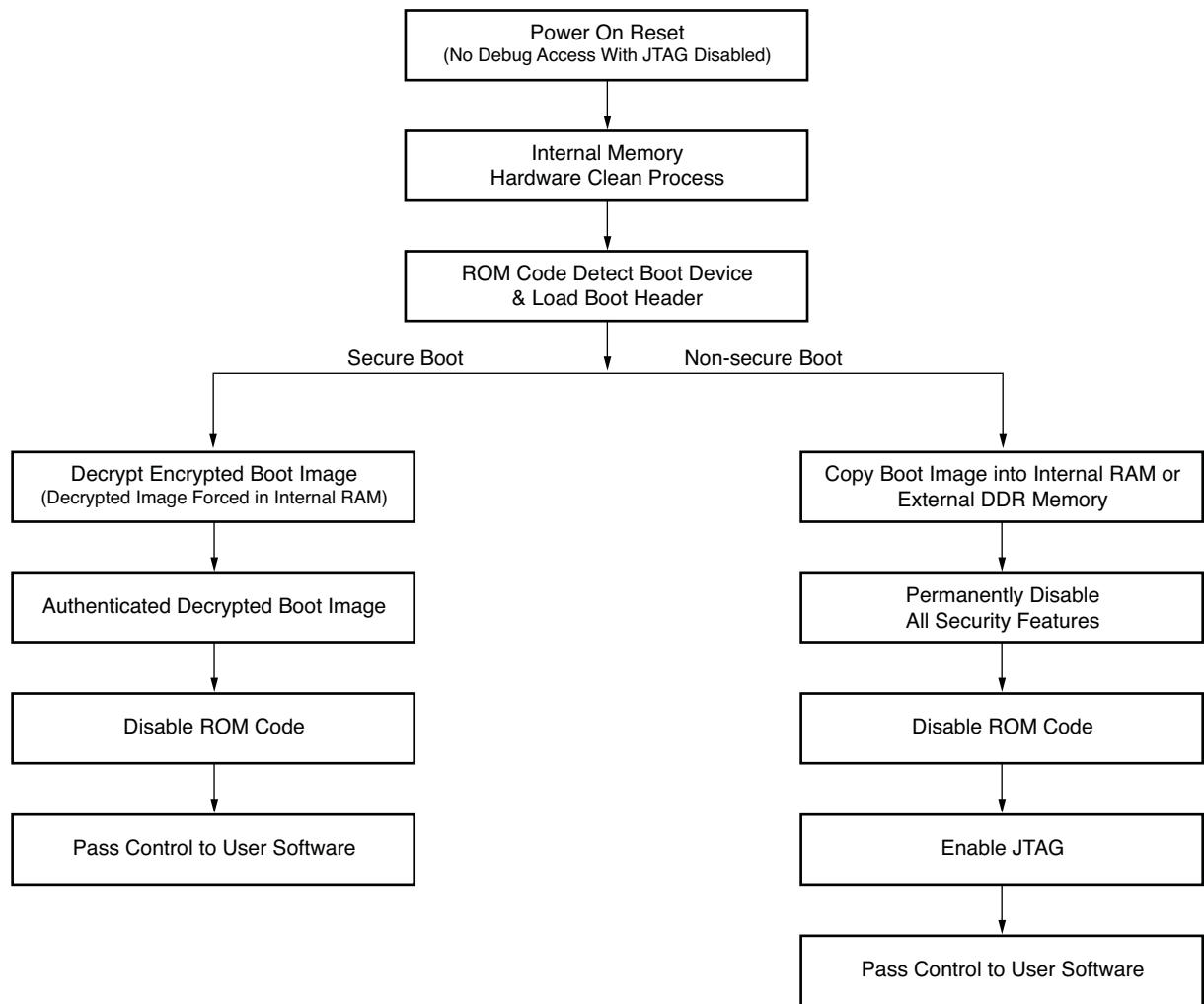
Figure 32-1: Secure Boot Block Diagram

A device secure boot involves several systems contained within the EPP device. The secure boot process is always initiated by the boot ROM. Once a secure boot has been detected, the boot ROM enables the AES and HMAC engines which reside in the PL. The encrypted first stage boot loader (FSBL) is fetched by the boot ROM and sent to the AES and HMAC engines in the PL via the processor configuration access port (PCAP). The FSBL image is decrypted and sent back to the PS via the PCAP where it is loaded into the on-chip RAM (OCM) for execution. The PS is then able to securely configure the PL. An encrypted bitstream is sent through the PCAP to the AES and the decrypted bitstream is then used to configure the PL.

32.2 Functional Description

32.2.1 Master Secure Boot

Master secure boot is the only secure boot mode supported in Zynq-7000 EPP devices. It uses the hardened AES decryption engine and the hardened HMAC authentication engine within the PL to decrypt PS images and PL bitstreams. The boot process for the master secure boot mode is shown in Figure 32-2.



UG585_c33_02_030412

Figure 32-2: PS Boot Flow

After the power-on and reset sequences have completed, the on-chip boot ROM begins to execute. The boot ROM starts by checking the boot mode specified by the bootstrap pins. The boot ROM then reads the boot header from the specified external memory.

If a secure boot is specified in the boot image header, the boot ROM starts by checking the power-on status of the PL. Because the AES and HMAC engines reside within the PL, the PL must be powered up to perform a secure boot. After the power-on status of the PL is confirmed, the boot ROM begins to load the encrypted FSBL into the AES engine via the PCAP. Once decrypted, the PL sends the plain text FSBL back to the PS via the PCAP. The decrypted image is then loaded into the OCM. The PS also monitors the authentication status of the PS image or PL bitstream. If an HMAC authentication error occurs, the PS enters security lock down.

Once the PS image has been successfully loaded and authenticated, control is turned over to the plain text FSBL which now resides in the OCM. Based on the user application, the FSBL could then either start processing, configure the PL, load additional software, or wait for further instruction from an external source.

The master secure boot mode uses the AES decryption and HMAC authentication engines within the PL, therefore the PL must be powered on during the secure boot process. The boot ROM ensures that the PL is powered before reading the encrypted image from the external boot device. If an additional encrypted PS image or PL bitstream is loaded at a later time, it is the users responsibly to ensure that the PL is powered on.

32.2.2 External Boot Devices

Secure boot mode is restricted to NOR, NAND, SDIO, or Quad-SPI flash as the external boot device. A secure boot from JTAG or any other external interface is not allowed.

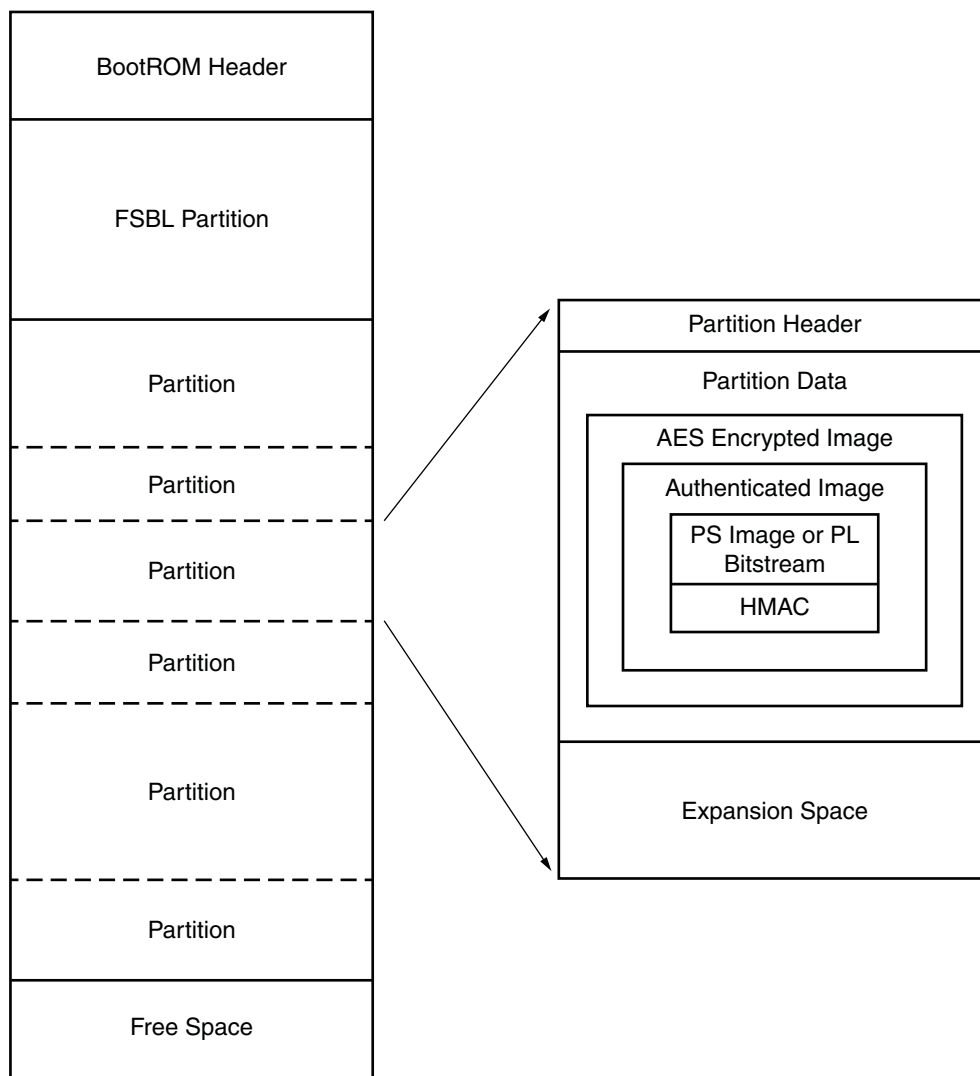
32.2.3 Secure Boot Image

The secure boot image format is show in [Figure 32-3](#) The secure boot image consists of a boot ROM header, a FSBL partition (required), and any number, including zero, of succeeding partitions. The boot ROM header identifies the boot image as secure or non-secure at offset 0x028. The value stored in the boot header at offset 0x028 determines the key source (see [Table 32-1](#)).

Table 32-1: Boot ROM Header Summary

Boot ROM Header Value at 0x028	Description
0xA5C3C5A3	Encrypted image using eFuse key
0x3A5C3C5A	Encrypted image using BBRAM key
All others	Non-encrypted image

The boot ROM header and partition headers are not encrypted. Partition data is signed and encrypted. The partition data is decrypted and authenticated by the AES and HMAC engines within the PL. When encryption is used, all partitions are encrypted. There cannot be a mix of encrypted and non-encrypted partitions in a boot image.



UG585_c33_03_030812

Figure 32-3: Secure Boot Image Format

32.2.4 eFuse Settings

The secure boot features can also be controlled via three PL eFuse bits that are described in [Table 32-2](#). (See [UG470](#), *7 Series FPGAs Configuration User Guide* for more information regarding eFuse.)

Table 32-2: eFuse Settings Summary

eFuse	Description
eFuse Secure Boot	The EPP device must boot securely and use the eFuse key as the AES key source. Non-secure boot of the device is not allowed. If the boot image header does not match this setting, a security lockdown occurs.
BBRAM Key Disable	If the EPP device is booted in secure mode, then the eFuse key must be selected. Non-secure boot of the device is allowed. If the boot image header does not match this setting, a security lockdown occurs.
JTAG Chain Disable	The ARM DAP controller is permanently set in bypass mode. Any attempt to activate the ARM DAP controller causes a security lockdown.

32.2.5 Boot Image and Bitstream Encryption

Boot images are assembled and encrypted using bootgen. A FSBL and any additional PS images or PL bitstreams along with the encryption key and authentication signature must be supplied to bootgen. The correct headers are generated automatically when bootgen builds the boot image.

32.2.6 Boot Image and Bitstream Decryption and Authentication

For cipher text decryption, Xilinx uses the advanced encryption standard (AES) in cipher block chaining (CBC) mode with a 256-bit key. PS images and PL bitstreams are authenticated with a keyed-hash message authentication code (HMAC) using the SHA-256 hash algorithm (FIPS180-2). When the Boot ROM detects that the FSBL image is encrypted, it enables the decryption and authentication engines within the PL. Both are enabled or disabled in tandem.

Subsequent PS images do not have to be encrypted. Once an encrypted FSBL has been loaded, it is “trusted” and can then load a plain text second stage boot loader or application. The path from the external memory to the OCM can bypass the decryption/authentication engines within the PL. Loading of plain text PS images after a secure boot is not recommended and should only be done after fully evaluating the system-level security.

32.2.7 HMAC Signature

The HMAC authentication requires a signature that must also be supplied to the bootgen software. This signature is not loaded into the PL directly via JTAG like the AES key. It is contained and protected by the encrypted PS image and the encrypted bitstream. During the on-chip decryption process, this HMAC signature is extracted from the image or bitstream and used by the authentication algorithm. No on-chip storage for the HMAC signature is required.

32.2.8 Key Management

The AES encryption key is stored on-chip within the PL. It can be loaded into either volatile battery-backed RAM (BBRAM) or in non-volatile eFuse storage. The keys are loaded into the PL via the JTAG interface using the iMPACT software. (See [UG470](#), *7 Series FPGAs Configuration User Guide* for more information)

32.3 Secure Boot Features

32.3.1 Non-Secure Boot State

The non-secure state is entered when the boot ROM detects that the FSBL is not encrypted. In this state the decryption and authentication engines are disabled and locked requiring a power-on reset (POR) to re-enable. All subsequent PS images, PL configuration bitstreams, and PL partial re-configuration bitstreams loaded via PCAP-SMAP, PCAP-ICAP, or ICAP must be unencrypted.

There is no mechanism to move from the non-secure state to the secure state, aside from power-on reset. Any attempt to load encrypted data after unencrypted data results in security violation and security lockdown.

32.3.2 Secure Boot State

The EPP is either operating in a secure state or a non-secure state. The secure state is entered when the boot ROM reads the encryption status from the boot ROM header section of the boot image. In this state the encrypted FSBL is loaded into the PS and any PL configuration using the PCAP-SMAP port must also be encrypted.

Because the encrypted FSBL loaded in a secure boot is “trusted”, it is possible to load additional PS images in plain text mode. PL partial re-configuration bitstreams can be loaded via the PCAP-ICAP or ICAP interface as either cipher text or plain text. Subsequent PS images or PL bitstreams must use the same key source as the FSBL, key switching is not allowed. Loading of plain text images or bitstreams after a secure boot is not recommended.

32.3.3 Security Lockdown

The PS's device configuration interface contains a security policy block that is used to monitor the system security. When conflicting status is detected either from the PS or the PL that could indicate inconsistent system configuration or tampering, a security lockdown is triggered. In a security lockdown the on-chip RAM is cleared along with all the system caches. The PL is reset and the PS enters a lockdown mode that can only be cleared by issuing a power-on reset. The following conditions cause a security reset:

- Non-secure boot and eFuse secure boot set
- PS DAP enabled and JTAG chain disable eFuse set

- SEU error tracking has been enabled in the PS and the PL reports an SEU error
- A discrepancy in the redundant AES enable logic
- Software sets the FORCE_RST bit of the Device Configuration Control register

32.3.4 Golden Image Search

The boot ROM supports the capability to fall-back and reload to a *golden* image (multi-boot). If the FSBL is plain text, the subsequent golden images must also be plain text. Loading an encrypted golden image after loading a plain text FSBL is not supported. Likewise, if the FSBL is cipher text, the golden image must also be encrypted with the same key as the FSBL. Loading a plain text golden image after loading a cipher text FSBL is not supported.

32.3.5 JTAG and Debug Considerations

In secure boot, the PS DAP and the PL TAP controllers are bypassed by default, eliminating any JTAG access to the EPP device. JTAG access can be restored in secure mode by the FSBL or subsequent PS images as these applications are considered *trusted*. Access to the DAP enable registers can be locked out using the Device Configuration Interface LOCK register.

The PS DAP controller can be permanently bypassed using the JTAG CHAIN DISABLE eFuse. The JTAG access to the PL can be disabled by setting the DISABLE_JTAG configuration option when creating the PL bitstream. (see [UG628, Command Line Tools User Guide](#) for more information.)

32.3.6 Readback

Whenever an encrypted bitstream is loaded into the PL, readback of the internal configuration memory cannot be performed by any of the external interfaces, including JTAG. The only readback access to the configuration memory after an encrypted bitstream load is via PCAP-ICAP or ICAP. The PCAP-ICAP and ICAP interfaces are *trusted* channels since access to these interfaces are from an authenticated PS image or an authenticated PL bitstream.

32.4 Programming Considerations

Although most of the secure boot process is handled by the boot ROM, it is possible to decrypt PS images or PL bitstreams after the initial boot. To decrypt secure images the PL must be powered on. The PL is powered on and ready to accept new encrypted data if the PCFG_INIT bit of the Device Configuration Interface (DevC) Status register is set.

To send encrypted data to the PL for decryption, the PCAP_MODE and PCAP_PR bits of the DEVC Control register must be set to 1. Because the AES engine decrypts data one byte at a time, the QUARTER_PCAP_RATE_EN bit on the DEVC Control register must also be set to 1.

To disable the AES and HMAC engines, the three PCFG_AES_EN bits of the DevC Control register must be set to 0. All three bits must be set with the same register write or a security violation occurs,

resulting in a security lockdown of the device. Once the AES and HMAC engines have been disabled, they cannot be enabled again without a power-on-reset.

Additional Resources

A.1 Xilinx Resources

Product Support and Documentation

www.xilinx.com/support

Xilinx Glossary

www.xilinx.com/company/terms.htm

Device User Guides

http://www.xilinx.com/support/documentation/zynq-7000_user_guides.htm

http://www.xilinx.com/support/documentation/7_series_user_guides.htm

Xilinx Design Tools: Installation and Licensing Guide (UG798)

http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/iil.pdf

Xilinx Design Tools: Release Notes Guide (UG631)

http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/irn.pdf

Xilinx Forums and Wiki Links

- <http://forums.xilinx.com>
- <http://wiki.xilinx.com>
- <http://wiki.xilinx.com/zynq-linux>
- <http://wiki.xilinx.com/zynq-uboot>

Xilinx git Websites

<http://git.xilinx.com>

A.2 Solution Centers

See the [Xilinx Solution Centers](#) for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

A.3 References

A.3.1 Zynq-7000 EPP Documents

Refer to the following Zynq-7000 EPP documents for further reference:

- CS692, *Zynq-7000 EPP Product Brief*
- DS190, *Zynq-7000 EPP Product Overview*
- DS187, *Zynq-7000 EPP (7z010 and 7z020): AC and DC Switching Characteristics Data Sheet*
- DS191, *Zynq-7000 EPP (7z030 and 7z045): AC and DC Switching Characteristics Data Sheet*
- UG865, *Zynq-7000 EPP Packaging and Pinout Specifications*
- UG821, *Zynq-7000 EPP Software Developers Guide*
- UG933, *Zynq-7000 EPP PCB Design and Pin Planning Guide*
- EN191, *Zynq-7000 EPP Errata Sheet*

A.3.2 PL Documents – Device and Boards

To learn more about the PL resources, refer to the following 7 Series FPGA User Guides:

- DS821, *Xilinx LogiCORE IP 7 Series FPGAs Integrated Block for PCI Express Product Specification*
- UG471, *Xilinx 7 Series FPGAs SelectIO Resources User Guide*
- UG472, *Xilinx 7 Series FPGAs Clocking Resources User Guide*
- UG473, *Xilinx 7 Series FPGAs Memory Resources User Guide*
- UG474, *Xilinx 7 Series FPGAs Configurable Logic Block User Guide*
- UG476, *Xilinx 7 Series FPGAs GTX Transceiver User Guide*
- UG477, *Xilinx 7 Series FPGAs Integrated Block v1.3 for PCI Express User Guide*
- UG479, *Xilinx 7 Series FPGAs DSP48E1 User Guide*
- UG480, *Xilinx 7 Series FPGAs XADC User Guide*
- UG483, *Xilinx 7-Series FPGAs PCB and Pin Planning Guide*

These user guides and additional relevant information can be found in the 7 Series overall set of documentation on the 7 Series FPGAs, which is available on the Xilinx website:

http://www.xilinx.com/support/documentation/7_series.htm

A.3.3 Advanced eXtensible Interface (AXI) Documents

Refer to [UG761](#), *AXI Reference Guide* for further reference on the AXI protocol.

A.3.4 Software Documents

Zynq-7000 EPP Software Developers Guide (UG821)

http://www.xilinx.com/support/documentation/zynq-7000_user_guides.htm

Zynq EDK Concepts, Tools, and Techniques Guide (UG873)

http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ug873-zynq-ctt.pdf

The source drivers for stand alone and FSBL are provided as part of the Xilinx IDE Design Suite Embedded Edition. The Linux drivers are provided via the Xilinx Open Source Wiki at:

<http://wiki.xilinx.com>

Xilinx Alliance Program partners provide system software solutions for IP, middleware, operation systems, etc. For the latest information refer to the Zynq-7000 landing page at:

<http://www.xilinx.com/products/silicon-devices/epp/zynq-7000>

A.3.5 git Information

- <http://git-scm.com>
- <http://git-scm.com/documentation>
- <http://git-scm.com/download>

A.3.6 Design Tool Documents

Xilinx Command Line Tools User Guide (UG628)

Xilinx ISE Manuals

http://www.xilinx.com/support/documentation/dt_ise.htm

Xilinx XPS/EDK Supported IP Website

http://www.xilinx.com/ise/embedded/edk_ip.htm

ChipScope Pro Software and Cores User Guide (UG029)

PlanAhead Tutorial: Debugging with ChipScope (UG677)

Xilinx Problem Solvers

<http://www.xilinx.com/support/troubleshoot.htm>

A.3.7 EDK Documentation

The entire EDK documentation set can be accessed at:

http://www.xilinx.com/support/documentation/dt_edk_edk14-1.htm

- *Embedded System Tools Reference Manual (UG111):*
http://www.xilinx.com/support/documentation/xilinx14_1/est_rm.pdf
- *OS and Libraries Document Collection (UG643):*
http://www.xilinx.com/support/documentation/xilinx14_1/oslib_rm.pdf
- *Platform Specification Format Reference Manual (UG642):*
http://www.xilinx.com/support/documentation/xilinx14_1/psf_rm.pdf

EDK Tutorials Website

http://www.xilinx.com/support/documentation/dt_edk_edk14-1_tutorials.htm

Platform Studio and EDK Website

http://www.xilinx.com/ise/embedded_design_prod/platform_studio.htm

XPS/EDK Supported IP Website

http://www.xilinx.com/ise/embedded/edk_ip.htm

A.3.8 Third-Party IP and Standards Documents

To learn about functional details related to vendor IP cores contained in Zynq-7000 devices or related international interface standards, refer the following documents:

Note: ARM documents can be found at: <http://infocenter.arm.com/help/index.jsp>

- *ARM AMBA Level 2 Cache Controller (L2C-310) TRM (also called PL310)*
- *ARM AMBA Specification Revision 2.0, 1999 (IHI 0011A)*
- *ARM Architecture Reference Manual (Need to register with ARM)*
- *ARM Cortex-A Series Programmer's Guide*

- *ARM Cortex-A9 Technical Reference Manual, Revision r3p0*
- *ARM Cortex-A9 MPCore Technical Reference Manual, Revision r3p0 (DDI0407F)* – includes descriptions for accelerator coherency port (ACP), CPU private timers and watchdog timers (AWDT), event bus, general interrupt controller (GIC), and snoop control unit (SCU)
- *ARM Cortex-A9 NEON Media Processing Engine Technical Reference Manual, Revision r3p0*
- *ARM Cortex-A9 Floating-Point Unit Technical Reference Manual, Revision r3p0*
- *ARM CoreSight v1.0 Architecture Specification* – includes descriptions for ATB Bus, and Authentication
- *ARM CoreSight Program Flow Trace Architecture Specification*
- *ARM Debug Interface v5.1 Architecture Specification*
- *ARM Debug Interface v5.1 Architecture Specification Supplement*
- *ARM CoreSight Components TRM* – includes descriptions for embedded cross trigger (ECT), embedded trace buffer (ETB), instrumentation trace macrocell (ITM), debug access port (DAP), and trace port interface unit (TPIU)
- *ARM CoreSight PTM-A9 TRM*
- *ARM CoreSight Trace Memory Controller Technical Reference Manual*
- *ARM Generic Interrupt Controller v1.0 Architecture Specification (IHI 0048B)*
- *ARM Generic Interrupt Controller PL390 Technical Reference Manual (DDI0416B)*
- *ARM PrimeCell DMA Controller (PL330) Technical Reference Manual*
- *ARM Application Note 239: Example programs for CoreLink DMA Controller DMA-330*
- *ARM PrimeCell Static Memory Controller (PL350 series) Technical Reference Manual, Revision r2p1, 12 October 2007 (ARM DDI 0380G)*
- *BOSCH, CAN Specification Version 2.0 PART A and PART B, 1991*
- *Cadence, Watchdog Timer (SWDT) Specification*
- *IEEE 802.3-2008 - IEEE Standard for Information technology-Specific requirements - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, 2008*
- *Intel Corp., Enhanced Host Controller Interface Specification for Universal Serial Bus, v1.0, 2002*
- *ISO 11898 Standard USB Association, USB 2.0 Specification*
- *SD Association, Part A2 SD Host Controller Standard Specification Ver2.00 Final 070130*
- *SD Association, Part E1 SDIO Specification Ver2.00 Final 070130*
- *SD Group, Part 1 Physical Layer Specification Ver2.00 Final 060509*

Register Details

B.1 Overview

This appendix provides details of all the memory-mapped registers in the Zynq™-7000 EPP.

Throughout this manual, the names of registers and register bit fields used match those given in the hardware. They are called the hardware names. C header files are delivered with this product which define register and bit field names for easy use in software code. In some cases, the software names are different from the hardware names. This appendix includes software names where they differ from hardware names:

- **Software Module Name:** This is included in the module introduction section and is named *Software Name*.
- **Software Register Name:** This is included in the detailed register description and is also named *Software Name*.
- **Software Bit field Name:** These are included in the register bit field tables in the *Field Name* column. These names follow the hardware field name and are contained in parentheses.

Note: If a software name does not exist in the C header files or if it exists but is the same as the hardware name, it is not included in this appendix and the above fields are not present.

The software register name will be:

`<software module name>_<software register name>_<optional suffix>`. One common suffix used for register names is "OFFSET", which would give the OFFSET address of that register from the base address for the module.

The software bit field name will be:

`<software module name>_<software register name>_<software bit field name>_<optional suffix>`. One common suffix used for bit field names is "MASK", which would be useful when extracting the bit field of interest from the full register.

B.2 Acronyms

The following acronyms are used for many of the registers.

Access Type	Description
clonrd	Readable, clears value on read
clonwr	Readable, clears value on write
nsnsro	During non-secure access, if thread is non-secure, it is read only
nsnsrw	During non-secure access, if thread is non-secure, it is read write
nsnswo	During non-secure access, if thread is non-secure, it is write only
nssraz	During non-secure access, if thread is secure, it is read as zero
raz	Read as zero
ro	Read-only
rud	Read undefined
rw	Normal read/write
rwso	Read/write, set only
sro	During secure access, it is read only
srw	During secure access, it is read write
swo	During secure access, it is write only
waz	Write as zero
wo	Write-only
wtc	Readable, write a one to clear
z	Access (read or write) as zero

B.3 Module Summary

Module Name	Module Type	Base Address	Version	Description
afi0	AFI	0xF8008000	1.5	AXI FIFO Interface Instance no. 0.
afi1	AFI	0xF8009000	1.5	AXI FIFO Interface Instance no. 1.
afi2	AFI	0xF800A000	1.5	AXI FIFO Interface Instance no. 2.
afi3	AFI	0xF800B000	1.5	AXI FIFO Interface Instance no. 3.
can0	can	0xE0008000	2.0.1	Controller Area Network Instance no. 0.
can1	can	0xE0009000	2.0.1	Controller Area Network Instance no. 1.
ddrc	ddrc	0xF8006000	1.4	DDR memory controller
debug_cpu_cti0	cti	0xF8898000	0.65	Cross Trigger Interface This cti instance is for CPU 0.
debug_cpu_cti1	cti	0xF8899000	0.65	Cross Trigger Interface This cti instance is for CPU 1.
debug_cpu_pmu0	cortexa9_pmu	0xF8891000	1.0	Cortex A9 Performance Monitoring Unit This pmu instance is for CPU 0.
debug_cpu_pmu1	cortexa9_pmu	0xF8893000	1.0	Cortex A9 Performance Monitoring Unit This pmu instance is for CPU 1.
debug_cpu_ptm0	ptm	0xF889C000	r1p0	CoreSight PTM-A9 This ptm instance is for CPU 0.
debug_cpu_ptm1	ptm	0xF889D000	r1p0	CoreSight PTM-A9 This ptm instance is for CPU 1.
debug_cti_etb_tpiu	cti	0xF8802000	0.65	Cross Trigger Interface This cti instance is for etb and tpiu.
debug_cti_ftm	cti	0xF8809000	0.65	Cross Trigger Interface This cti instance is for ftm.
debug_dap_rom	dap	0xF8800000	0.65	Debug Access Port ROM Table
debug_etb	etb	0xF8801000	0.65	Embedded Trace Buffer
debug_ftm	ftm	0xF880B000	0.65	Fabric Trace Macrocell
debug_funnel	funnel	0xF8804000	0.65	CoreSight Trace Funnel

Module Name	Module Type	Base Address	Version	Description
debug_itm	itm	0xF8805000	0.65	Instrumentation Trace Macrocell
debug_tpiu	tpiu	0xF8803000	0.65	Trace Port Interface Unit
devcfg	devcfg	0xF8007000	1.3	Device configuraion Interface
dmac0_ns	dmac	0xF8004000	R1P1	direct memory access controller, PL330 This address space is used when the DMAC is in non-secure mode.
dmac0_s	dmac	0xF8003000	R1P1	direct memory access controller, PL330 This address space is used when the DMAC is in secure mode.
gem0	GEM	0xE000B000	1.0	Gigabit Ethernet Controller Instance no. 0.
gem1	GEM	0xE000C000	1.0	Gigabit Ethernet Controller Instance no. 1.
gpio	gpio	0xE000A000	1.4	General Purpose Input / Output
gpv_qos301_cpu	qos301	0xF8946000	r0p0	The AMBA Network Interconnect Advanced Quality of Service (QoS-301) is an extension to the AMBA Network Interconnect (NIC-301) base product and provides programmable QoS facilities for attached AMBA masters. CPU-to-DDR port instance.
gpv_qos301_dmac	qos301	0xF8947000	r0p0	The AMBA Network Interconnect Advanced Quality of Service (QoS-301) is an extension to the AMBA Network Interconnect (NIC-301) base product and provides programmable QoS facilities for attached AMBA masters. DMAC port instance.
gpv_qos301_iou	qos301	0xF8948000	r0p0	The AMBA Network Interconnect Advanced Quality of Service (QoS-301) is an extension to the AMBA Network Interconnect (NIC-301) base product and provides programmable QoS facilities for attached AMBA masters. IOU port instance.
gpv_trustzone	nic301_addr_region_ctrl_registers	0xF8900000	1.0	AMBA Network Interconnect NIC301, Address Region Control Registers NIC301 TrustZone.
i2c0	IIC	0xE0004000	1.0	Inter Integrated Circuit (I2C) Instance no. 0.
i2c1	IIC	0xE0005000	1.0	Inter Integrated Circuit (I2C) Instance no. 1.
l2cache	L2Cpl310	0xF8F02000	r3p2v	L2 cache PL310

Module Name	Module Type	Base Address	Version	Description
mpcore	mpcore	0xF8F00000	r2p2	Mpcore - SCU, Interrupt controller, Counters and Timers
ocm	ocm	0xF800C000	1.0	On-Chip Memory Registers
qspi	qspi	0xE000D000	1.0	LQSPI module Registers
sd0	sdio	0xE0100000	4.4a	SD2.0/ SDIO2.0/ MMC3.31 AHB Host ControllerRegisters Instance no. 0.
sd1	sdio	0xE0101000	4.4a	SD2.0/ SDIO2.0/ MMC3.31 AHB Host ControllerRegisters Instance no. 1.
slcr	slcr	0xF8000000	1.0	System Level Control Registers
smcc	pl353	0xE000E000	1.0	Shared memory controller
spi0	SPI	0xE0006000	1.0	Serial Peripheral Interface Instance no. 0.
spi1	SPI	0xE0007000	1.0	Serial Peripheral Interface Instance no. 1.
swdt	swdt	0xF8005000	7.0	System Watchdog Timer Registers
ttc0	ttc	0xF8001000	5.0	Triple Timer Counter Instance no. 0.
ttc1	ttc	0xF8002000	5.0	Triple Timer Counter Instance no. 1.
uart0	UART	0xE0000000	1.0	Universal Asynchronous Receiver Transmitter Instance no. 0.
uart1	UART	0xE0001000	1.0	Universal Asynchronous Receiver Transmitter Instance no. 1.
usb0	usb	0xE0002000	2.2	USB controller registers Instance no. 0.
usb1	usb	0xE0003000	2.2	USB controller registers Instance no. 1.

50 modules, 2344 registers.

B.4 AXI FIFO Interface (AFI)

Module Name	AXI FIFO Interface (AFI)
Base Address	0xF8008000 afi0 0xF8009000 afi1 0xF800A000 afi2 0xF800B000 afi3
Description	AXI FIFO Interface Instance no. 0.
Version	1.5
Doc Version	1.1
Vendor Info	Xilinx AFI

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
AFI_RDCHAN_CTRL	0x00000000	32	mixed	0x00000000	Read Channel Control Register: Control fields for Read Channel operation
AFI_RDCHAN_ISSUINGCAP	0x00000004	32	mixed	0x00000007	Read Issuing Capability Register: Sets the maximum number of Outstanding Read Commands allowed (Issuing Capability). Refers to the commands that can be outstanding from the AFI to the SAM switch and back. Fields are selected by the 'axds_rdisuecap1_en' input.
AFI_RDQOS	0x00000008	32	mixed	0x00000000	QOS Read Channel Register: Sets the static Qos value to be used for the read channel. If APB register field, 'FabricQosEn' is 0 or ('FabricQosEn' is 1 and 'QosHeadOfCmdQEn' is 1), this static Qos value will be applied to all read commands enqueued into the RdCmdQ. If ('FabricQosEn' is 1 and 'QosHeadOfCmdQEn' is 0), this static Qos field will be ignored.

Register Name	Address	Width	Type	Reset Value	Description
AFI_RDDATAFIFO_LEVEL	0x0000000C	32	mixed	0x00000000	Read Data FIFO Level Register: Returns the Level of the Read Data FIFO in Qwords. Note that this register should only be read if a valid HP port clock is actively running. If no clock is running the APB access will hang.
AFI_RDDEBUG	0x00000010	32	mixed	0x00000000	Read Channel Debug Register: Miscellaneous debug fields for the Read channel. Not to be used for functional purposes.
AFI_WRCHAN_CTRL	0x00000014	32	mixed	0x00000F00	Write Channel Control Register: Control fields for Write Channel operation
AFI_WRCHAN_ISSUINGCAP	0x00000018	32	mixed	0x00000007	Write Issuing Capability Register: Sets the maximum number of Outstanding Write Commands (Issuing Capability) allowed. Refers to the commands that can be outstanding from the AFI to the SAM switch and back. Fields are selected by the 'axds_wrissuecap1_en' input.
AFI_WRQOS	0x0000001C	32	mixed	0x00000000	QOS Write Channel Register: Sets the static Qos value to be used for the write channel. If APB register field, 'FabricQosEn' is 0 or ('FabricQosEn' is 1 and 'QosHeadOfCmdQEn' is 1), this static Qos value will be applied to all write commands enqueued into the WrCmdQ. If ('FabricQosEn' is 1 and 'QosHeadOfCmdQEn' is 0), this static Qos field will be ignored.
AFI_WRDATAFIFO_LEVEL	0x00000020	32	mixed	0x00000000	Write Data FIFO Level Register: Returns the Level of the Write Data FIFO in Dwords. Note that this register should only be read if a valid HP port clock is actively running. If no clock is present, the APB access will hang.
AFI_WRDEBUG	0x00000024	32	mixed	0x00000000	Write Channel Debug Register

Register (AFI) AFI_RDCHAN_CTRL

Name	AFI_RDCHAN_CTRL
Relative Address	0x00000000
Absolute Address	afi0: 0xF8008000 afi1: 0xF8009000 afi2: 0xF800A000 afi3: 0xF800B000
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Read Channel Control Register: Control fields for Read Channel operation

Register AFI_RDCHAN_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:4	raz	0x0	Return 0 when read
QosHeadOfCmdQEn	3	rw	0x0	When set, allows the priority of a transaction at the head of the RdCmdQ to be promoted if higher priority transactions are backed up behind it. The entire RdCmdQ will therefore be promoted when the fabric RdQos signal is promoted. When disabled, only the new read commands issued will receive the promotion.
FabricOutCmdEn	2	rw	0x0	Enable control of outstanding read commands from the fabric 0 - The maximum number of outstanding read commands is always taken from APB register field, rdIssueCap0 1 - The maximum outstanding number of read commands is selected from the fabric input, axds_rdisuecap1_en, as follows: Max Outstanding Read Commands = axds_rdisuecap1_en ? rdIssueCap1 : rdIssueCap0
FabricQosEn	1	rw	0x0	Enable control of qos from the fabric 0 - The qos bits are derived from APB register, AFI_RDQOS.staticQos 1 - The qos bits are dynamically driven from the fabric input, axds_arqos[3:0]
32BitEn	0	rw	0x0	Configures the Read Channel as a 32-bit interface. 1 - 32-bit enabled 0 - 64-bit enabled

Register (AFI) AFI_RDCHAN_ISSUINGCAP

Name	AFI_RDCHAN_ISSUINGCAP
Relative Address	0x00000004
Absolute Address	afi0: 0xF8008004 afi1: 0xF8009004 afi2: 0xF800A004 afi3: 0xF800B004
Width	32 bits
Access Type	mixed
Reset Value	0x00000007
Description	Read Issuing Capability Register: Sets the maximum number of Outstanding Read Commands allowed (Issuing Capability). Refers to the commands that can be outstanding from the AFI to the SAM switch and back. Fields are selected by the 'axds_rdisuecap1_en' input.

Register AFI_RDCHAN_ISSUINGCAP Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:7	raz	0x0	Return 0 when read
rdIssueCap1	6:4	rw	0x0	Max number of outstanding read commands (Read Issuing Capability) field 1: 3'b000: 1 command 3'b001: 2 commands' '3'b111: 8 commands
reserved	3	raz	0x0	Return 0 when read
rdIssueCap0	2:0	rw	0x7	Max number of outstanding read commands (Read Issuing Capability) field 0: 3'b000: 1 command 3'b001: 2 commands' '3'b111: 8 commands

Register (AFI) AFI_RDQOS

Name	AFI_RDQOS
Relative Address	0x00000008
Absolute Address	afi0: 0xF8008008 afi1: 0xF8009008 afi2: 0xF800A008 afi3: 0xF800B008
Width	32 bits
Access Type	mixed
Reset Value	0x00000000

Description QOS Read Channel Register: Sets the static Qos value to be used for the read channel. If APB register field, 'FabricQosEn' is 0 or ('FabricQosEn' is 1 and 'QosHeadOfCmdQEn' is 1), this static Qos value will be applied to all read commands enqueued into the RdCmdQ. If ('FabricQosEn' is 1 and 'QosHeadOfCmdQEn' is 0), this static Qos field will be ignored.

Register AFI_RDQOS Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:4	raz	0x0	Return 0 when read
staticQos	3:0	rw	0x0	Sets the level of the Qos field to be used for the read channel 4'b0000: Lowest Priority ' '4'b1111: Highest Priority

Register ([AFI](#)) AFI_RDDATAFIFO_LEVEL

Name AFI_RDDATAFIFO_LEVEL

Relative Address 0x0000000C

Absolute Address afi0: 0xF800800C
afi1: 0xF800900C
afi2: 0xF800A00C
afi3: 0xF800B00C

Width 32 bits

Access Type mixed

Reset Value 0x00000000

Description Read Data FIFO Level Register:Returns the Level of the Read Data FIFO in Qwords. Note that this register should only be read if a valid HP port clock is actively running. If no clock is running the APB access will hang.

Register AFI_RDDATAFIFO_LEVEL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	raz	0x0	Return 0 when read
FifoLevel	7:0	ro	0x0	Returns the level measured in Dwords (64-bits) of the Read Data FIFO 8'h00 - 0 Entries 8'h01 - 1 Entry' '8'h8F - 128 Entries

Register ([AFI](#)) AFI_RDDEBUG

Name AFI_RDDEBUG

Relative Address 0x00000010

Absolute Address	afi0: 0xF8008010 afi1: 0xF8009010 afi2: 0xF800A010 afi3: 0xF800B010
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Read Channel Debug Register: Miscellaneous debug fields for the Read channel. Not to be used for functional purposes.

Register AFI_RDDEBUG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:5	raz	0x0	Return 0 when read
OutRdCmds	4:1	ro	0x0	Returns the number of read commands in flight between the AFI and the SAM switch 4'h0: 0 4'h1: 1 etc
RdDataFifoOverflow	0	ro	0x0	Bit is set if the RdDataFIFO overflows

Register ([AFI](#)) AFI_WRCHAN_CTRL

Name	AFI_WRCHAN_CTRL
Relative Address	0x00000014
Absolute Address	afi0: 0xF8008014 afi1: 0xF8009014 afi2: 0xF800A014 afi3: 0xF800B014
Width	32 bits
Access Type	mixed
Reset Value	0x00000F00
Description	Write Channel Control Register: Control fields for Write Channel operation

Register AFI_WRCHAN_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:12	raz	0x0	Return 0 when read
WrDataThreshold	11:8	rw	0xF	<p>Sets the threshold at which to send the write command. Note that this is measured in data beats, and is therefore dependent on the '32bitEn' field.</p> <p>4'b0000: Send Write Command When 1 data beat is pushed into the Write Data FIFO</p> <p>4'b0001: Send Write Command When 2 data beats are pushed into the Write Data FIFO</p> <p>4'b1111: Send Write Command When 16 data beats are pushed into the Write Data FIFO</p> <p>Note: If this field is programmed to be less than the actual burst length of the write command, the 'Wlast' will take priority. For example, if 'WrDataThreshold' is set to 4'b1111 (indicates 16 beats), and a Wlast is received after 8 beats, the write command is sent.</p>
reserved	7:6	raz	0x0	Return 0 when read
WrCmdReleaseMode	5:4	rw	0x0	<p>Mode of Write Command Release.</p> <p>2'b00: Release Wr Command on 'Wlast' enqueue into Write Data FIFO</p> <p>2'b01: Release Wr Command on a particular threshold being reached on the enqueue into Write Data FIFO. The 'WrDataThreshold' field is used to program the actual threshold.</p> <p>2'b10: Release Wr Command immediately it is received. If we use this mode actively, then switch to another mode, we must issue a (hard/soft) reset in between.</p> <p>2'b11: Reserved</p>
QosHeadOfCmdQEn	3	rw	0x0	<p>When set, allows the priority of a transaction at the head of the WrCmdQ to be promoted if higher priority transactions are backed up behind it. The entire WrCmdQ will therefore be 'promoted' when the fabric 'WrQos' signal is promoted.</p> <p>When disabled, only the new write commands issued will receive the 'promotion'.</p>

Field Name	Bits	Type	Reset Value	Description
FabricOutCmdEn	2	rw	0x0	Enable control of outstanding write commands from the fabric 0 - The maximum number of outstanding write commands is always taken from APB register field, 'wrIssueCap0' 1 - The maximum outstanding number of write commands is selected from the fabric input, 'axds_wrissuecap1_en', as follows: Max Outstanding Write Commands = axds_wrissuecap1_en ? wrIssueCap1 : wrIssueCap0
FabricQosEn	1	rw	0x0	Enable control of qos from the fabric 0 - The qos bits are derived from APB register, 'AFI_WRQOS.staticQos' 1 - The qos bits are dynamically driven from the fabric input, 'axds_awqos[3:0]'
32BitEn	0	rw	0x0	Configures the Write Channel as a 32-bit interface. 1 - 32-bit enabled 0 - 64-bit enabled

Register ([AFI](#)) AFI_WRCHAN_ISSUINGCAP

Name	AFI_WRCHAN_ISSUINGCAP
Relative Address	0x00000018
Absolute Address	afi0: 0xF8008018 afi1: 0xF8009018 afi2: 0xF800A018 afi3: 0xF800B018
Width	32 bits
Access Type	mixed
Reset Value	0x00000007
Description	Write Issuing Capability Register: Sets the maximum number of Outstanding Write Commands (Issuing Capability) allowed. Refers to the commands that can be outstanding from the AFI to the SAM switch and back. Fields are selected by the 'axds_wrissuecap1_en' input.

Register AFI_WRCHAN_ISSUINGCAP Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:7	raz	0x0	Return 0 when read
wrIssueCap1	6:4	rw	0x0	Max number of outstanding write commands (Write Issuing Capability) field 1: 3'b000: 1 command 3'b001: 2 commands' '3'b111: 8 commands
reserved	3	raz	0x0	Return 0 when read
wrIssueCap0	2:0	rw	0x7	Max number of outstanding write commands (Write Issuing Capability) field 0: 3'b000: 1 command 3'b001: 2 commands' '3'b111: 8 commands

Register ([AFI](#)) AFI_WRQOS

Name	AFI_WRQOS
Relative Address	0x0000001C
Absolute Address	afi0: 0xF800801C afi1: 0xF800901C afi2: 0xF800A01C afi3: 0xF800B01C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	QOS Write Channel Register: Sets the static Qos value to be used for the write channel. If APB register field, 'FabricQosEn' is 0 or ('FabricQosEn' is 1 and 'QosHeadOfCmdQEn' is 1), this static Qos value will be applied to all write commands enqueued into the WrCmdQ. If ('FabricQosEn' is 1 and 'QosHeadOfCmdQEn' is 0), this static Qos field will be ignored.

Register AFI_WRQOS Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:4	raz	0x0	Return 0 when read
staticQos	3:0	rw	0x0	Sets the level of the Qos field to be used for the write channel 4'b0000: Lowest Priority' '4'b1111: Highest Priority

Register (AFI) AFI_WRDATAFIFO_LEVEL

Name	AFI_WRDATAFIFO_LEVEL
Relative Address	0x00000020
Absolute Address	afi0: 0xF8008020 afi1: 0xF8009020 afi2: 0xF800A020 afi3: 0xF800B020
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Write Data FIFO Level Register: Returns the Level of the Write Data FIFO in Dwords. Note that this register should only be read if a valid HP port clock is actively running. If no clock is present, the APB access will hang.

Register AFI_WRDATAFIFO_LEVEL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	raz	0x0	Return 0 when read
FifoLevel	7:0	ro	0x0	Returns the level measured in Dwords (64-bits) of the Write Data FIFO 8'h00 - 0 Entries 8'h01 - 1 Entry' '8'h8F - 128 Entries

Register (AFI) AFI_WRDEBUG

Name	AFI_WRDEBUG
Relative Address	0x00000024
Absolute Address	afi0: 0xF8008024 afi1: 0xF8009024 afi2: 0xF800A024 afi3: 0xF800B024
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Write Channel Debug Register

Register AFI_WRDEBUG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:5	raz	0x0	Return 0 when read
OutWrCmds	4:1	ro	0x0	Returns the number of write commands in flight between the AFI and the SAM switch 4'h0: 0 4'h1: 1 etc
WrDataFifoOverflow	0	ro	0x0	Bit is set if the WrDataFIFO overflows

B.5 CAN Controller (can)

Module Name	CAN Controller (can)
Software Name	XCANPS
Base Address	0xE0008000 can0 0xE0009000 can1
Description	Controller Area Network Instance no. 0.
Version	2.0.1
Doc Version	1.0
Vendor Info	Xilinx CAN

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
SRR	0x00000000	32	rw	0x00000000	Software Reset Register
MSR	0x00000004	32	rw	0x00000000	Mode Select Register
BRPR	0x00000008	32	rw	0x00000000	Baud Rate Prescaler Register
BTR	0x0000000C	32	rw	0x00000000	Bit Timing Register
ECR	0x00000010	32	ro	0x00000000	Error Counter Register
ESR	0x00000014	32	mixed	0x00000000	Error Status Register
SR	0x00000018	32	mixed	0x00000001	Status Register
ISR	0x0000001C	32	mixed	0x00006000	Interrupt Status Register
IER	0x00000020	32	rw	0x00000000	Interrupt Enable Register
ICR	0x00000024	32	mixed	0x00000000	Interrupt Clear Register
TCR	0x00000028	32	mixed	0x00000000	Timestamp Control Register
WIR	0x0000002C	32	rw	0x00003F3F	Watermark Interrupt Register
TXFIFO_ID	0x00000030	32	wo	0x00000000	transmit message fifo message identifier
TXFIFO_DLC	0x00000034	32	rw	0x00000000	transmit message fifo data length code
TXFIFO_DATA1	0x00000038	32	rw	0x00000000	transmit message fifo data word 1
TXFIFO_DATA2	0x0000003C	32	rw	0x00000000	transmit message fifo data word 2

Register Name	Address	Width	Type	Reset Value	Description
TXHPB_ID	0x00000040	32	wo	0x00000000	transmit high priority buffer message identifier
TXHPB_DLC	0x00000044	32	rw	0x00000000	transmit high priority buffer data length code
TXHPB_DATA1	0x00000048	32	rw	0x00000000	transmit high priority buffer data word 1
TXHPB_DATA2	0x0000004C	32	rw	0x00000000	transmit high priority buffer data word 2
RXFIFO_ID	0x00000050	32	ro	x	receive message fifo message identifier
RXFIFO_DLC	0x00000054	32	rw	x	receive message fifo data length code
RXFIFO_DATA1	0x00000058	32	rw	x	receive message fifo data word 1
RXFIFO_DATA2	0x0000005C	32	rw	x	receive message fifo data word 2
AFR	0x00000060	32	rw	0x00000000	Acceptance Filter Register
AFMR1	0x00000064	32	rw	x	Acceptance Filter Mask Register 1
AFIR1	0x00000068	32	rw	x	Acceptance Filter ID Register 1
AFMR2	0x0000006C	32	rw	x	Acceptance Filter Mask Register 2
AFIR2	0x00000070	32	rw	x	Acceptance Filter ID Register 2
AFMR3	0x00000074	32	rw	x	Acceptance Filter Mask Register 3
AFIR3	0x00000078	32	rw	x	Acceptance Filter ID Register 3
AFMR4	0x0000007C	32	rw	x	Acceptance Filter Mask Register 4
AFIR4	0x00000080	32	rw	x	Acceptance Filter ID Register 4

Register ([can](#)) SRR

Name	SRR
Relative Address	0x00000000
Absolute Address	can0: 0xE0008000 can1: 0xE0009000
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Software Reset Register

Register SRR Details

Writing to the Software Reset Register (SRR) places the CAN controller in Configuration mode. Once in Configuration mode, the CAN controller drives recessive on the bus line and does not transmit or receive messages. During power-up, CEN and SRST bits are '0' and CONFIG bit in the Status Register (SR) is '1.' The Transfer Layer Configuration Registers can be changed only when CEN bit in the SRR Register is '0.' If the CEN bit is changed during core operation, it is recommended to reset the core so that operations start afresh.

Field Name	Bits	Type	Reset Value	Description
reserved	31:2	rw	0x0	Reserved.
CEN	1	rw	0x0	<p>Can Enable</p> <p>The Enable bit for the CAN controller.</p> <p>1 = The CAN controller is in Loop Back, Sleep or Normal mode depending on the LBACK and SLEEP bits in the MSR.</p> <p>0 = The CAN controller is in the Configuration mode.</p> <p>If the CEN bit is changed during core operation, it is recommended to reset the core so that operations start afresh.</p>
SRST	0	rw	0x0	<p>Reset</p> <p>The Software reset bit for the CAN controller.</p> <p>1 = CAN controller is reset.</p> <p>If a 1 is written to this bit, all the CAN controller configuration registers (including the SRR) are reset. Reads to this bit always return a 0.</p>

Register ([can](#)) MSR

Name	MSR
Relative Address	0x00000004
Absolute Address	can0: 0xE0008004 can1: 0xE0009004
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Mode Select Register

Register MSR Details

Writing to the Mode Select Register (MSR) enables the CAN controller to enter Snoop, Sleep, Loop Back, or Normal modes. In Normal mode, the CAN controller participates in normal bus communication. If the SLEEP bit is set to '1,' the CAN controller enters Sleep mode. If the LBACK bit is set to '1,' the CAN controller enters Loop Back mode. If the SNOOP mode is set to '1,' the CAN controller enters Snoop mode and does not participate in bus communication but only receives messages.

The LBACK and SLEEP bits should never be set to '1' at the same time. At any given point the CAN controller can be either in Loop Back mode or Sleep mode, but not both simultaneously. If both bits are set, then LBACK Mode takes priority. SNOOP Mode has least priority. In order for core to enter SNOOP mode LBACK and SLEEP modes should be set to '0'.

Field Name	Bits	Type	Reset Value	Description
reserved	31:3	rw	0x0	Reserved.
SNOOP	2	rw	0x0	Snoop Mode Select The Snoop Mode Select bit. 1 = CAN controller is in Snoop mode. 0 = CAN controller is in Normal, Loop Back, Configuration, or Sleep mode. This bit can be written to only when CEN bit in SRR is 0.
LBACK	1	rw	0x0	Loop Back Mode Select The Loop Back Mode Select bit. 1 = CAN controller is in Loop Back mode. 0 = CAN controller is in Normal, Snoop, Configuration, or Sleep mode. This bit can be written to only when CEN bit in SRR is 0.
SLEEP	0	rw	0x0	Sleep Mode Select The Sleep Mode select bit. 1 = CAN controller is in Sleep mode. 0 = CAN controller is in Normal, Snoop, Configuration or Loop Back mode. This bit is cleared when the CAN controller wakes up from the Sleep mode.

Register ([can](#)) BRPR

Name	BRPR
Relative Address	0x00000008
Absolute Address	can0: 0xE0008008 can1: 0xE0009008
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Baud Rate Prescaler Register

Register BRPR Details

The BRPR can be programmed to any value in the range 0-255. The actual value is 1 more than the value written into the register.

Please refer to the CAN chapter for more details.

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rw	0x0	Reserved
BRP	7:0	rw	0x0	Baud Rate Prescaler These bits indicate the prescaler value. The actual value ranges from 1 to 256.

Register ([can](#)) BTR

Name	BTR
Relative Address	0x0000000C
Absolute Address	can0: 0xE000800C can1: 0xE000900C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Bit Timing Register

Register BTR Details

The Bit Timing Register (BTR) specifies the bits needed to configure bit time. Specifically, the Propagation Segment, Phase segment 1, Phase segment 2, and Synchronization Jump Width (as defined in CAN 2.0A, CAN 2.0B and ISO 11891-1) are written to the BTR. The actual value of each of these fields is one more than the value written to this register.

Field Name	Bits	Type	Reset Value	Description
reserved	31:9	rw	0x0	Reserved
SJW	8:7	rw	0x0	Synchronization Jump Width Indicates the Synchronization Jump Width as specified in the CAN 2.0A and CAN 2.0B standard. The actual value is one more than the value written to the register.

Field Name	Bits	Type	Reset Value	Description
TS2	6:4	rw	0x0	Time Segment 2 Indicates Phase Segment 2 as specified in the CAN 2.0A and CAN 2.0B standard. The actual value is one more than the value written to the register.
TS1	3:0	rw	0x0	Time Segment 1 Indicates the Sum of Propagation Segment and Phase Segment 1 as specified in the CAN 2.0A and CAN 2.0B standard. The actual value is one more than the value written to the register.

Register ([can](#)) ECR

Name	ECR
Relative Address	0x00000010
Absolute Address	can0: 0xE0008010 can1: 0xE0009010
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Error Counter Register

Register ECR Details

The ECR is a read-only register. Writes to the ECR have no effect. The value of the error counters in the register reflect the values of the transmit and receive error counters in the CAN Protocol Engine Module (see Figure 1).

The following conditions reset the Transmit and Receive Error counters:

- * When '1' is written to the SRST bit in the SRR
- * When '0' is written to the CEN bit in the SRR
- * When the CAN controller enters Bus Off state
- * During Bus Off recovery when the CAN controller enters Error Active state after 128 occurrences of 11 consecutive recessive bits

When in Bus Off recovery, the Receive Error counter is advanced by 1 when a sequence of 11 consecutive recessive bits is seen.

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved
REC	15:8	ro	0x0	Receive Error Counter Indicates the Value of the Receive Error Counter.
TEC	7:0	ro	0x0	Transmit Error Counter Indicates the Value of the Transmit Error Counter.

Register ([can](#)) ESR

Name	ESR
Relative Address	0x00000014
Absolute Address	can0: 0xE0008014 can1: 0xE0009014
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Error Status Register

Register ESR Details

The Error Status Register (ESR) indicates the type of error that has occurred on the bus. If more than one error occurs, all relevant error flag bits are set in this register. The ESR is a write-to-clear register. Writes to this register will not set any bits, but will clear the bits that are set.

Field Name	Bits	Type	Reset Value	Description
reserved	31:5	rw	0x0	Reserved
ACKER	4	wtc	0x0	ACK Error Indicates an acknowledgment error. 1 = Indicates an acknowledgment error has occurred. 0 = Indicates an acknowledgment error has not occurred on the bus since the last write to this register. If this bit is set, writing a 1 clears it.
BERR	3	wtc	0x0	Bit Error Indicates the received bit is not the same as the transmitted bit during bus communication. 1 = Indicates a bit error has occurred. 0 = Indicates a bit error has not occurred on the bus since the last write to this register. If this bit is set, writing a 1 clears it.

Field Name	Bits	Type	Reset Value	Description
STER	2	wtc	0x0	<p>Stuff Error</p> <p>Indicates an error if there is a stuffing violation.</p> <p>1 = Indicates a stuff error has occurred.</p> <p>0 = Indicates a stuff error has not occurred on the bus since the last write to this register.</p> <p>If this bit is set, writing a 1 clears it.</p>
FMER	1	wtc	0x0	<p>Form Error</p> <p>Indicates an error in one of the fixed form fields in the message frame.</p> <p>1 = Indicates a form error has occurred.</p> <p>0 = Indicates a form error has not occurred on the bus since the last write to this register.</p> <p>If this bit is set, writing a 1 clears it.</p>
CRCER	0	wtc	0x0	<p>CRC Error</p> <p>Indicates a CRC error has occurred.</p> <p>1 = Indicates a CRC error has occurred.</p> <p>0 = Indicates a CRC error has not occurred on the bus since the last write to this register.</p> <p>If this bit is set, writing a 1 clears it.</p> <p>In case of a CRC Error and a CRC delimiter corruption, only the FMER bit is set.</p>

Register ([can](#)) SR

Name	SR
Relative Address	0x00000018
Absolute Address	can0: 0xE0008018 can1: 0xE0009018
Width	32 bits
Access Type	mixed
Reset Value	0x00000001
Description	Status Register

Register SR Details

The CAN Status Register provides a status of all conditions of the Core. Specifically, FIFO status, Error State, Bus State and Configuration mode are reported.

Field Name	Bits	Type	Reset Value	Description
reserved	31:13	rw	0x0	Reserved
SNOOP	12	ro	0x0	Snoop Mode Indicates the CAN controller is in Snoop Mode. 1 = Indicates the CAN controller is in Snoop Mode. 0 = Indicates the CAN controller is not in Snoop mode.
ACFBSY	11	ro	0x0	Acceptance Filter Busy This bit indicates that the Acceptance Filter Mask Registers and the Acceptance Filter ID Registers cannot be written to. 1 = Acceptance Filter Mask Registers and Acceptance Filter ID Registers cannot be written to. 0 = Acceptance Filter Mask Registers and the Acceptance Filter ID Registers can be written to. This bit exists only when the number of acceptance filters is not 0 This bit is set when a 0 is written to any of the valid UAF bits in the Acceptance Filter Register.
TXFLL	10	ro	0x0	Transmit FIFO Full Indicates that the TX FIFO is full. 1 = Indicates the TX FIFO is full. 0 = Indicates the TX FIFO is not full.
TXBFLL	9	ro	0x0	High Priority Transmit Buffer Full Indicates the High Priority Transmit Buffer is full. 1 = Indicates the High Priority Transmit Buffer is full. 0 = Indicates the High Priority Transmit Buffer is not full.
ESTAT	8:7	ro	0x0	Error Status Indicates the error status of the CAN controller. 00 = Indicates Configuration Mode (CONFIG = 1). Error State is undefined. 01 = Indicates Error Active State. 11 = Indicates Error Passive State. 10 = Indicates Bus Off State.

Field Name	Bits	Type	Reset Value	Description
ERRWRN	6	ro	0x0	<p>Error Warning</p> <p>Indicates that either the Transmit Error counter or the Receive Error counter has exceeded a value of 96.</p> <p>1 = One or more error counters have a value greater than or equal to 96.</p> <p>0 = Neither of the error counters has a value greater than or equal to 96.</p>
BBSY	5	ro	0x0	<p>Bus Busy</p> <p>Indicates the CAN bus status.</p> <p>1 = Indicates that the CAN controller is either receiving a message or transmitting a message.</p> <p>0 = Indicates that the CAN controller is either in Configuration mode or the bus is idle.</p>
BIDLE	4	ro	0x0	<p>Bus Idle</p> <p>Indicates the CAN bus status.</p> <p>1 = Indicates no bus communication is taking place.</p> <p>0 = Indicates the CAN controller is either in Configuration mode or the bus is busy.</p>
NORMAL	3	ro	0x0	<p>Normal Mode</p> <p>Indicates the CAN controller is in Normal Mode.</p> <p>1 = Indicates the CAN controller is in Normal Mode.</p> <p>0 = Indicates the CAN controller is not in Normal mode.</p>
SLEEP	2	ro	0x0	<p>Sleep Mode</p> <p>Indicates the CAN controller is in Sleep mode.</p> <p>1 = Indicates the CAN controller is in Sleep mode.</p> <p>0 = Indicates the CAN controller is not in Sleep mode.</p>

Field Name	Bits	Type	Reset Value	Description
LBACK	1	ro	0x0	<p>Loop Back Mode</p> <p>Indicates the CAN controller is in Loop Back mode.</p> <p>1 = Indicates the CAN controller is in Loop Back mode.</p> <p>0 = Indicates the CAN controller is not in Loop Back mode.</p>
CONFIG	0	ro	0x1	<p>Configuration Mode Indicator</p> <p>Indicates the CAN controller is in Configuration mode.</p> <p>1 = Indicates the CAN controller is in Configuration mode.</p> <p>0 = Indicates the CAN controller is not in Configuration mode.</p>

Register ([can](#)) ISR

Name	ISR
Relative Address	0x0000001C
Absolute Address	can0: 0xE000801C can1: 0xE000901C
Width	32 bits
Access Type	mixed
Reset Value	0x00006000
Description	Interrupt Status Register

Register ISR Details

The Interrupt Status Register (ISR) contains bits that are set when a particular interrupt condition occurs. If the corresponding mask bit in the Interrupt Enable Register is set, an interrupt is generated.

Interrupt bits in the ISR can be cleared by writing to the Interrupt Clear Register. For all bits in the ISR, a set condition takes priority over the clear condition and the bit continues to remain '1.'

Field Name	Bits	Type	Reset Value	Description
reserved	31:15	rw	0x0	reserved
TXFEMP (IXR_TXFEMP)	14	ro	0x1	<p>Transmit FIFO EmptyInterrupt</p> <p>A 1 indicates that the Transmit FIFO is empty.</p> <p>The interrupt continues to assert as long as the TX FIFO is empty.</p> <p>This bit can be cleared only by writing to the ICR.</p>

Field Name	Bits	Type	Reset Value	Description
TXFWMEMP (IXR_TXFWMEMP)	13	ro	0x1	Transmit FIFO Watermark Empty Interrupt A 1 indicates that the TX FIFO is empty based on watermark programming. The interrupt continues to assert as long as the number of empty spaces in the TX FIFO is greater than TX FIFO empty watermark. This bit can be cleared only by writing to the Interrupt Clear Register.
RXFWMFLL (IXR_RXFWMFLL)	12	ro	0x0	Receive FIFO Watermark Full Interrupt A 1 indicates that the RX FIFO is full based on watermark programming. The interrupt continues to assert as long as the RX FIFO count is above RX FIFO Full watermark. This bit can be cleared only by writing to the Interrupt Clear Register.
WKUP (IXR_WKUP)	11	ro	0x0	Wake up Interrupt A 1 indicates that the CAN controller entered Normal mode from Sleep Mode. This bit can be cleared by writing to the ICR. This bit is also cleared when a 0 is written to the CEN bit in the SRR.
SLP (IXR_SLP)	10	ro	0x0	Sleep Interrupt A 1 indicates that the CAN controller entered Sleep mode. This bit can be cleared by writing to the ICR. This bit is also cleared when a 0 is written to the CEN bit in the SRR.
BSOFF (IXR_BSOFF)	9	ro	0x0	Bus Off Interrupt A 1 indicates that the CAN controller entered the Bus Off state. This bit can be cleared by writing to the ICR. This bit is also cleared when a 0 is written to the CEN bit in the SRR.

Field Name	Bits	Type	Reset Value	Description
ERROR (IXR_ERROR)	8	ro	0x0	<p>Error Interrupt</p> <p>A 1 indicates that an error occurred during message transmission or reception.</p> <p>This bit can be cleared by writing to the ICR.</p> <p>This bit is also cleared when a 0 is written to the CEN bit in the SRR.</p>
RXNEMP (IXR_RXNEMP)	7	ro	0x0	<p>Receive FIFO Not Empty Interrupt</p> <p>A 1 indicates that the Receive FIFO is not empty.</p> <p>This bit can be cleared only by writing to the ICR.</p>
RXOFLW (IXR_RXOFLW)	6	ro	0x0	<p>RX FIFO Overflow Interrupt</p> <p>A 1 indicates that a message has been lost. This condition occurs when a new message is being received and the Receive FIFO is Full.</p> <p>This bit can be cleared by writing to the ICR.</p> <p>This bit is also cleared when a 0 is written to the CEN bit in the SRR.</p>
RXUFLW (IXR_RXUFLW)	5	ro	0x0	<p>RX FIFO Underflow Interrupt</p> <p>A 1 indicates that a read operation was attempted on an empty RX FIFO.</p> <p>This bit can be cleared only by writing to the ICR.</p>
RXOK (IXR_RXOK)	4	ro	0x0	<p>New Message Received Interrupt</p> <p>A 1 indicates that a message was received successfully and stored into the RX FIFO.</p> <p>This bit can be cleared by writing to the ICR.</p> <p>This bit is also cleared when a 0 is written to the CEN bit in the SRR.</p>
TXBFLL (IXR_TXBFLL)	3	ro	0x0	<p>High Priority Transmit Buffer Full Interrupt</p> <p>A 1 indicates that the High Priority Transmit Buffer is full.</p> <p>The status of the bit is unaffected if write transactions occur on the High Priority Transmit Buffer when it is already full.</p> <p>This bit can be cleared only by writing to the ICR.</p>
TXFLL (IXR_TXFLL)	2	ro	0x0	<p>Transmit FIFO Full Interrupt</p> <p>A 1 indicates that the TX FIFO is full.</p> <p>The status of the bit is unaffected if write transactions occur on the Transmit FIFO when it is already full.</p> <p>This bit can be cleared only by writing to the Interrupt Clear Register.</p>

Field Name	Bits	Type	Reset Value	Description
TXOK (IXR_TXOK)	1	ro	0x0	<p>Transmission Successful Interrupt</p> <p>A 1 indicates that a message was transmitted successfully.</p> <p>This bit can be cleared by writing to the ICR.</p> <p>This bit is also cleared when a 0 is written to the CEN bit in the SRR.</p> <p>In Loop Back mode, both TXOK and RXOK bits are set. The RXOK bit is set before the TXOK bit.</p>
ARBLST (IXR_ARBLST)	0	ro	0x0	<p>Arbitration Lost Interrupt</p> <p>A 1 indicates that arbitration was lost during message transmission.</p> <p>This bit can be cleared by writing to the ICR.</p> <p>This bit is also cleared when a 0 is written to the CEN bit in the SRR.</p>

Register ([can](#)) IER

Name	IER
Relative Address	0x00000020
Absolute Address	can0: 0xE0008020 can1: 0xE0009020
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Interrupt Enable Register

Register IER Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:15	rw	0x0	Reserved
ETXFEMP (IXR_TXFEMP)	14	rw	0x0	<p>Enable TXFIFO Empty Interrupt</p> <p>Writes to this bit enable or disable interrupts when the</p> <p>TXFEMP bit in the ISR is set.</p> <p>1 = Enable interrupt generation if TXFEMP bit in ISR is set.</p> <p>0 = Disable interrupt generation if TXFEMP bit in ISR is set.</p>

Field Name	Bits	Type	Reset Value	Description
ETXFWMEMP (IXR_TXFWMEMP)	13	rw	0x0	Enable TXFIFO watermark Empty Interrupt Writes to this bit enable or disable interrupts when the TXFWMEMP bit in the ISR is set. 1 = Enable interrupt generation if TXFWMEMP bit in ISR is set. 0 = Disable interrupt generation if TXFWMEMP bit in ISR is set.
ERXFWMFLL (IXR_RXFWMFLL)	12	rw	0x0	Enable RXFIFO watermark Full Interrupt Writes to this bit enable or disable interrupts when the RXFLL bit in the ISR is set. 1 = Enable interrupt generation if RXFWMFLL bit in ISR is set. 0 = Disable interrupt generation if RXFWMFLL bit in ISR is set.
EWKUP (IXR_WKUP)	11	rw	0x0	Enable Wake up Interrupt Writes to this bit enable or disable interrupts when the WKUP bit in the ISR is set. 1 = Enable interrupt generation if WKUP bit in ISR is set. 0 = Disable interrupt generation if WKUP bit in ISR is set.
ESLP (IXR_SLP)	10	rw	0x0	Enable Sleep Interrupt Writes to this bit enable or disable interrupts when the SLP bit in the ISR is set. 1 = Enable interrupt generation if SLP bit in ISR is set. 0 = Disable interrupt generation if SLP bit in ISR is set.
EBSOFF (IXR_BSOFF)	9	rw	0x0	Enable Bus OFF Interrupt Writes to this bit enable or disable interrupts when the BSOFF bit in the ISR is set. 1 = Enable interrupt generation if BSOFF bit in ISR is set. 0 = Disable interrupt generation if BSOFF bit in ISR is set.

Field Name	Bits	Type	Reset Value	Description
EERROR (IXR_ERROR)	8	rw	0x0	Enable Error Interrupt Writes to this bit enable or disable interrupts when the ERROR bit in the ISR is set. 1 = Enable interrupt generation if ERROR bit in ISR is set. 0 = Disable interrupt generation if ERROR bit in ISR is set.
ERXNEMP (IXR_RXNEMP)	7	rw	0x0	Enable Receive FIFO Not Empty Interrupt Writes to this bit enable or disable interrupts when the RXNEMP bit in the ISR is set. 1 = Enable interrupt generation if RXNEMP bit in ISR is set. 0 = Disable interrupt generation if RXNEMP bit in ISR is set.
ERXOFLW (IXR_RXOFLW)	6	rw	0x0	Enable RX FIFO Overflow Interrupt Writes to this bit enable or disable interrupts when the RXOFLW bit in the ISR is set. 1 = Enable interrupt generation if RXOFLW bit in ISR is set. 0 = Disable interrupt generation if RXOFLW bit in ISR is set.
ERXUFLW (IXR_RXUFLW)	5	rw	0x0	Enable RX FIFO Underflow Interrupt Writes to this bit enable or disable interrupts when the RXUFLW bit in the ISR is set. 1 = Enable interrupt generation if RXUFLW bit in ISR is set. 0 = Disable interrupt generation if RXUFLW bit in ISR is set.

Field Name	Bits	Type	Reset Value	Description
ERXOK (IXR_RXOK)	4	rw	0x0	Enable New Message Received Interrupt Writes to this bit enable or disable interrupts when the RXOK bit in the ISR is set. 1 = Enable interrupt generation if RXOK bit in ISR is set. 0 = Disable interrupt generation if RXOK bit in ISR is set.
ETXBFLl (IXR_TXBFLl)	3	rw	0x0	Enable High Priority Transmit Buffer Full Interrupt Writes to this bit enable or disable interrupts when the TXBFLl bit in the ISR is set. 1 = Enable interrupt generation if TXBFLl bit in ISR is set. 0 = Disable interrupt generation if TXBFLl bit in ISR is set.
ETXFLL (IXR_TXFLL)	2	rw	0x0	Enable Transmit FIFO Full Interrupt Writes to this bit enable or disable interrupts when TXFLL bit in the ISR is set. 1 = Enable interrupt generation if TXFLL bit in ISR is set. 0 = Disable interrupt generation if TXFLL bit in ISR is set.

Field Name	Bits	Type	Reset Value	Description
ETXOK (IXR_TXOK)	1	rw	0x0	Enable Transmission Successful Interrupt Writes to this bit enable or disable interrupts when the TXOK bit in the ISR is set. 1 = Enable interrupt generation if TXOK bit in ISR is set. 0 = Disable interrupt generation if TXOK bit in ISR is set.
EARBLST (IXR_ARBLST)	0	rw	0x0	Enable Arbitration Lost Interrupt Writes to this bit enable or disable interrupts when the ARBLST bit in the ISR is set. 1 = Enable interrupt generation if ARBLST bit in ISR is set. 0 = Disable interrupt generation if ARBLST bit in ISR is set.

Register ([can](#)) ICR

Name	ICR
Relative Address	0x00000024
Absolute Address	can0: 0xE0008024 can1: 0xE0009024
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Interrupt Clear Register

Register ICR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:15	rw	0x0	Reserved
CTXFEMP (IXR_TXFEMP)	14	wo	0x0	Clear TXFIFO Empty Interrupt Writing a 1 to this bit clears the TXFEMP bit in the ISR.

Field Name	Bits	Type	Reset Value	Description
CTXFWMEMP (IXR_TXFWMEMP)	13	wo	0x0	Clear TXFIFO Watermark Empty Interrupt Writing a 1 to this bit clears the TXFWMEMP bit in the ISR.
CRXFWMFLL (IXR_RXFWMFLL)	12	wo	0x0	Clear RXFIFO Watermark Full Interrupt Writing a 1 to this bit clears the RXFWMFLL bit in the ISR.
CWKUP (IXR_WKUP)	11	wo	0x0	Clear Wake up Interrupt Writing a 1 to this bit clears the WKUP bit in the ISR.
CSLP (IXR_SLP)	10	wo	0x0	Clear Sleep Interrupt Writing a 1 to this bit clears the SLP bit in the ISR.
CBSOFF (IXR_BSOFF)	9	wo	0x0	Clear Bus Off Interrupt Writing a 1 to this bit clears the BSOFF bit in the ISR.
CERROR (IXR_ERROR)	8	wo	0x0	Clear Error Interrupt Writing a 1 to this bit clears the ERROR bit in the ISR.
CRXNEMP (IXR_RXNEMP)	7	wo	0x0	Clear Receive FIFO Not Empty Interrupt Writing a 1 to this bit clears the RXNEMP bit in the ISR.
CRXOFLW (IXR_RXOFLW)	6	wo	0x0	Clear RX FIFO Overflow Interrupt Writing a 1 to this bit clears the RXOFLW bit in the ISR.
CRXUFLW (IXR_RXUFLW)	5	wo	0x0	Clear RX FIFO Underflow Interrupt Writing a 1 to this bit clears the RXUFLW bit in the ISR.
CRXOK (IXR_RXOK)	4	wo	0x0	Clear New Message Received Interrupt Writing a 1 to this bit clears the RXOK bit in the ISR.
CTXBFLL (IXR_TXBFLL)	3	wo	0x0	Clear High Priority Transmit Buffer Full Interrupt Writing a 1 to this bit clears the TXBFLL bit in the ISR.
CTXFLL (IXR_TXFLL)	2	wo	0x0	Clear Transmit FIFO Full Interrupt Writing a 1 to this bit clears the TXFLL bit in the ISR.

Field Name	Bits	Type	Reset Value	Description
CTXOK (IXR_TXOK)	1	wo	0x0	Clear Transmission Successful Interrupt Writing a 1 to this bit clears the CTXOK bit in the ISR.
CARBLST (IXR_ARBLST)	0	wo	0x0	Clear Arbitration Lost Interrupt Writing a 1 to this bit clears the ARBLST bit in the ISR.

Register ([can](#)) TCR

Name	TCR
Relative Address	0x00000028
Absolute Address	can0: 0xE0008028 can1: 0xE0009028
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Timestamp Control Register

Register TCR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	reserved
CTS	0	wo	0x0	Clear Timestamp Internal free running counter is cleared to 0 when CTS=1. This bit only needs to be written once with a 1 to clear the counter. The bit will automatically return to 0.

Register ([can](#)) WIR

Name	WIR
Relative Address	0x0000002C
Absolute Address	can0: 0xE000802C can1: 0xE000902C
Width	32 bits
Access Type	rw
Reset Value	0x00003F3F

Description Watermark Interrupt Register

Register WIR Details

The TXFIFO Empty watermark (EW) programmed in this register will be applied to TX FIFO only. The RXFIFO Full watermark (FW) programmed in this register will be applied to RX FIFO only. The watermark register is allowed to program only when CEN=0 in SRR register.

The TXFIFO Watermark EMPTY interrupt (TXFWMEMP) will continue to assert as long as the number of empty spaces in the TX FIFO is greater than the TXFIFO Empty watermark (EW). The RXFLL interrupt will continue to assert as long as the RX FIFO count remains above the RXFIFO Full watermark (FW).

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	rw	0x0	reserved
EW	15:8	rw	0x3F	TXFIFO Empty watermark TXFIFO generates an EMPTY interrupt based on the value programmed in this field. The valid range is (1-63). Only bits 18-23 are writable. No protection is given for illegal programming in this field. This field can be written to only when CEN bit in SRR is 0.
FW	7:0	rw	0x3F	RXFIFO Full watermark RXFIFO generates FULL interrupt based on the value programmed in this field. The valid range is (1-63). Only bits 26-31 are writable. No protection is given for illegal programming in this field. This field can be written to only when CEN bit in SRR is 0.

Register ([can](#)) TXFIFO_ID

Name	TXFIFO_ID
Relative Address	0x00000030
Absolute Address	can0: 0xE0008030 can1: 0xE0009030
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	transmit message fifo message identifier

Register TXFIFO_ID Details

Field Name	Bits	Type	Reset Value	Description
IDH (IDR_ID1)	31:21	wo	0x0	Standard Message ID The Identifier portion for a Standard Frame is 11 bits. These bits indicate the Standard Frame ID. This field is valid for both Standard and Extended Frames.
SRRRTR (IDR_SRR)	20	wo	0x0	Substitute Remote Transmission Request This bit differentiates between data frames and remote frames. Valid only for Standard Frames. For Extended frames this bit is 1. 1 = Indicates that the message frame is a Remote Frame. 0 = Indicates that the message frame is a Data Frame.
IDE (IDR_IDE)	19	wo	0x0	Identifier Extension This bit differentiates between frames using the Standard Identifier and those using the Extended Identifier. Valid for both Standard and Extended Frames. 1 = Indicates the use of an Extended Message Identifier. 0 = Indicates the use of a Standard Message Identifier.
IDL (IDR_ID2)	18:1	wo	0x0	Extended Message ID This field indicates the Extended Identifier. Valid only for Extended Frames. For Standard Frames, reads from this field return 0s For Standard Frames, writes to this field should be 0s
RTR (IDR_RTR)	0	wo	0x0	Remote Transmission Request This bit differentiates between data frames and remote frames. Valid only for Extended Frames. 1 = Indicates the message object is a Remote Frame 0 = Indicates the message object is a Data Frame For Standard Frames, reads from this bit returns 0 For Standard Frames, writes to this bit should be 0

Register ([can](#)) TXFIFO_DLC

Name	TXFIFO_DLC
Relative Address	0x00000034
Absolute Address	can0: 0xE0008034 can1: 0xE0009034
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	transmit message fifo data length code

Register TXFIFO_DLC Details

Field Name	Bits	Type	Reset Value	Description
DLC (DLCR_DLC)	31:28	rw	0x0	Data Length Code This is the data length portion of the control field of the CAN frame. This indicates the number valid data bytes in Data Word 1 and Data Word 2 registers.
reserved	27:0	rw	0x0	reserved

Register ([can](#)) TXFIFO_DATA1

Name	TXFIFO_DATA1
Software Name	TXFIFO_DW1
Relative Address	0x00000038
Absolute Address	can0: 0xE0008038 can1: 0xE0009038
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	transmit message fifo data word 1

Register TXFIFO_DATA1 Details

Field Name	Bits	Type	Reset Value	Description
DB0 (DW1R_DB0)	31:24	rw	0x0	Data Byte 0 Reads from this field return invalid data if the message has no data.
DB1 (DW1R_DB1)	23:16	rw	0x0	Data Byte 1 Reads from this field return invalid data if the message has only 1 byte of data or fewer
DB2 (DW1R_DB2)	15:8	rw	0x0	Data Byte 2 Reads from this field return invalid data if the message has only 2 byte of data or fewer
DB3 (DW1R_DB3)	7:0	rw	0x0	Data Byte 3 Reads from this field return invalid data if the message has only 3 byte of data or fewer

Register ([can](#)) TXFIFO_DATA2

Name	TXFIFO_DATA2
Software Name	TXFIFO_DW2
Relative Address	0x0000003C
Absolute Address	can0: 0xE000803C can1: 0xE000903C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	transmit message fifo data word 2

Register TXFIFO_DATA2 Details

Field Name	Bits	Type	Reset Value	Description
DB0 (DW2R_DB4)	31:24	rw	0x0	Data Byte 4 Reads from this field return invalid data if the message has only 4 byte of data or fewer
DB1 (DW2R_DB5)	23:16	rw	0x0	Data Byte 5 Reads from this field return invalid data if the message has only 5 byte of data or fewer

Field Name	Bits	Type	Reset Value	Description
DB2 (DW2R_DB6)	15:8	rw	0x0	Data Byte 6 Reads from this field return invalid data if the message has only 6 byte of data or fewer
DB3 (DW2R_DB7)	7:0	rw	0x0	Data Byte 7 Reads from this field return invalid data if the message has 7 byte of data or fewer

Register ([can](#)) TXHPB_ID

Name	TXHPB_ID
Relative Address	0x00000040
Absolute Address	can0: 0xE0008040 can1: 0xE0009040
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	transmit high priority buffer message identifier

Register TXHPB_ID Details

Field Name	Bits	Type	Reset Value	Description
IDH (IDR_ID1)	31:21	wo	0x0	Standard Message ID The Identifier portion for a Standard Frame is 11 bits. These bits indicate the Standard Frame ID. This field is valid for both Standard and Extended Frames.
SRRRTR (IDR_SRR)	20	wo	0x0	Substitute Remote Transmission Request This bit differentiates between data frames and remote frames. Valid only for Standard Frames. For Extended frames this bit is 1. 1 = Indicates that the message frame is a Remote Frame. 0 = Indicates that the message frame is a Data Frame.

Field Name	Bits	Type	Reset Value	Description
IDE (IDR_IDE)	19	wo	0x0	Identifier Extension This bit differentiates between frames using the Standard Identifier and those using the Extended Identifier. Valid for both Standard and Extended Frames. 1 = Indicates the use of an Extended Message Identifier. 0= Indicates the use of a Standard Message Identifier.
IDL (IDR_ID2)	18:1	wo	0x0	Extended Message ID This field indicates the Extended Identifier. Valid only for Extended Frames. For Standard Frames, reads from this field return 0s For Standard Frames, writes to this field should be 0s
RTR (IDR_RTR)	0	wo	0x0	Remote Transmission Request This bit differentiates between data frames and remote frames. Valid only for Extended Frames. 1 = Indicates the message object is a Remote Frame 0 = Indicates the message object is a Data Frame For Standard Frames, reads from this bit returns 0 For Standard Frames, writes to this bit should be 0

Register ([can](#)) TXHPB_DLC

Name	TXHPB_DLC
Relative Address	0x00000044
Absolute Address	can0: 0xE0008044 can1: 0xE0009044
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	transmit high priority buffer data length code

Register TXHPB_DLC Details

Field Name	Bits	Type	Reset Value	Description
DLC (DLCR_DLC)	31:28	rw	0x0	Data Length Code This is the data length portion of the control field of the CAN frame. This indicates the number valid data bytes in Data Word 1 and Data Word 2 registers.
reserved	27:0	rw	0x0	reserved

Register ([can](#)) TXHPB_DATA1

Name	TXHPB_DATA1
Software Name	TXHPB_DW1
Relative Address	0x00000048
Absolute Address	can0: 0xE0008048 can1: 0xE0009048
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	transmit high priority buffer data word 1

Register TXHPB_DATA1 Details

Field Name	Bits	Type	Reset Value	Description
DB0 (DW1R_DB0)	31:24	rw	0x0	Data Byte 0 Reads from this field return invalid data if the message has no data.
DB1 (DW1R_DB1)	23:16	rw	0x0	Data Byte 1 Reads from this field return invalid data if the message has only 1 byte of data or fewer
DB2 (DW1R_DB2)	15:8	rw	0x0	Data Byte 2 Reads from this field return invalid data if the message has only 2 byte of data or fewer
DB3 (DW1R_DB3)	7:0	rw	0x0	Data Byte 3 Reads from this field return invalid data if the message has only 3 byte of data or fewer

Register ([can](#)) TXHPB_DATA2

Name	TXHPB_DATA2
Software Name	TXHPB_DW2
Relative Address	0x0000004C
Absolute Address	can0: 0xE000804C can1: 0xE000904C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	transmit high priority buffer data word 2

Register TXHPB_DATA2 Details

Field Name	Bits	Type	Reset Value	Description
DB0 (DW2R_DB4)	31:24	rw	0x0	Data Byte 4 Reads from this field return invalid data if the message has only 4 byte of data or fewer
DB1 (DW2R_DB5)	23:16	rw	0x0	Data Byte 5 Reads from this field return invalid data if the message has only 5 byte of data or fewer
DB2 (DW2R_DB6)	15:8	rw	0x0	Data Byte 6 Reads from this field return invalid data if the message has only 6 byte of data or fewer
DB3 (DW2R_DB7)	7:0	rw	0x0	Data Byte 7 Reads from this field return invalid data if the message has 7 byte of data or fewer

Register ([can](#)) RXFIFO_ID

Name	RXFIFO_ID
Relative Address	0x00000050
Absolute Address	can0: 0xE0008050 can1: 0xE0009050
Width	32 bits
Access Type	ro
Reset Value	x
Description	receive message fifo message identifier

Register RXFIFO_ID Details

Field Name	Bits	Type	Reset Value	Description
IDH (IDR_ID1)	31:21	ro	x	Standard Message ID The Identifier portion for a Standard Frame is 11 bits. These bits indicate the Standard Frame ID. This field is valid for both Standard and Extended Frames.
SRRRTR (IDR_SRR)	20	ro	x	Substitute Remote Transmission Request This bit differentiates between data frames and remote frames. Valid only for Standard Frames. For Extended frames this bit is 1. 1 = Indicates that the message frame is a Remote Frame. 0 = Indicates that the message frame is a Data Frame.
IDE (IDR_IDE)	19	ro	x	Identifier Extension This bit differentiates between frames using the Standard Identifier and those using the Extended Identifier. Valid for both Standard and Extended Frames. 1 = Indicates the use of an Extended Message Identifier. 0 = Indicates the use of a Standard Message Identifier.
IDL (IDR_ID2)	18:1	ro	x	Extended Message ID This field indicates the Extended Identifier. Valid only for Extended Frames. For Standard Frames, reads from this field return 0s For Standard Frames, writes to this field should be 0s
RTR (IDR_RTR)	0	ro	x	Remote Transmission Request This bit differentiates between data frames and remote frames. Valid only for Extended Frames. 1 = Indicates the message object is a Remote Frame 0 = Indicates the message object is a Data Frame For Standard Frames, reads from this bit returns 0 For Standard Frames, writes to this bit should be 0

Register ([can](#)) RXFIFO_DLC

Name	RXFIFO_DLC
Relative Address	0x00000054
Absolute Address	can0: 0xE0008054 can1: 0xE0009054
Width	32 bits
Access Type	rw
Reset Value	x
Description	receive message fifo data length code

Register RXFIFO_DLC Details

Field Name	Bits	Type	Reset Value	Description
DLC (DLCR_DLC)	31:28	rw	x	Data Length Code This is the data length portion of the control field of the CAN frame. This indicates the number valid data bytes in Data Word 1 and Data Word 2 registers.
reserved	27:16	rw	x	reserved
RXT	15:0	rw	x	RX Timestamp

Register ([can](#)) RXFIFO_DATA1

Name	RXFIFO_DATA1
Software Name	RXFIFO_DW1
Relative Address	0x00000058
Absolute Address	can0: 0xE0008058 can1: 0xE0009058
Width	32 bits
Access Type	rw
Reset Value	x
Description	receive message fifo data word 1

Register RXFIFO_DATA1 Details

Field Name	Bits	Type	Reset Value	Description
DB0 (DW1R_DB0)	31:24	rw	x	Data Byte 0 Reads from this field return invalid data if the message has no data.
DB1 (DW1R_DB1)	23:16	rw	x	Data Byte 1 Reads from this field return invalid data if the message has only 1 byte of data or fewer
DB2 (DW1R_DB2)	15:8	rw	x	Data Byte 2 Reads from this field return invalid data if the message has only 2 byte of data or fewer
DB3 (DW1R_DB3)	7:0	rw	x	Data Byte 3 Reads from this field return invalid data if the message has only 3 byte of data or fewer

Register ([can](#)) RXFIFO_DATA2

Name	RXFIFO_DATA2
Software Name	RXFIFO_DW2
Relative Address	0x0000005C
Absolute Address	can0: 0xE000805C can1: 0xE000905C
Width	32 bits
Access Type	rw
Reset Value	x
Description	receive message fifo data word 2

Register RXFIFO_DATA2 Details

Field Name	Bits	Type	Reset Value	Description
DB0 (DW2R_DB4)	31:24	rw	x	Data Byte 4 Reads from this field return invalid data if the message has only 4 byte of data or fewer
DB1 (DW2R_DB5)	23:16	rw	x	Data Byte 5 Reads from this field return invalid data if the message has only 5 byte of data or fewer

Field Name	Bits	Type	Reset Value	Description
DB2 (DW2R_DB6)	15:8	rw	x	Data Byte 6 Reads from this field return invalid data if the message has only 6 byte of data or fewer
DB3 (DW2R_DB7)	7:0	rw	x	Data Byte 7 Reads from this field return invalid data if the message has 7 byte of data or fewer

Register ([can](#)) AFR

Name	AFR
Relative Address	0x00000060
Absolute Address	can0: 0xE0008060 can1: 0xE0009060
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Acceptance Filter Register

Register AFR Details

The Acceptance Filter Register (AFR) defines which acceptance filters to use. Each Acceptance Filter ID Register (AFIR) and Acceptance Filter Mask Register (AFMR) pair is associated with a UAF bit.

When the UAF bit is '1,' the corresponding acceptance filter pair is used for acceptance filtering. When the UAF bit is '0,' the corresponding acceptance filter pair is not used for acceptance filtering.

To modify an acceptance filter pair in Normal mode, the corresponding UAF bit in this register must be set to '0.' After the acceptance filter is modified, the corresponding UAF bit must be set to '1.' The following conditions govern the number of UAF bits that can exist in the.

* If all UAF bits are set to '0,' then all received messages are stored in the RX FIFO

* If the UAF bits are changed from a '1' to '0' during reception of a CAN message, the message may not be stored in the RX FIFO.

Field Name	Bits	Type	Reset Value	Description
reserved	31:4	rw	0x0	reserved
UAF4	3	rw	0x0	Use Acceptance Filter Number 4 Enables the use of acceptance filter pair 4. 1 = Indicates Acceptance Filter Mask Register 4 and Acceptance Filter ID Register 4 are used for acceptance filtering. 0 = Indicates Acceptance Filter Mask Register 4 and Acceptance Filter ID Register 4 are not used for acceptance filtering.
UAF3	2	rw	0x0	Use Acceptance Filter Number 3 Enables the use of acceptance filter pair 3. 1 = Indicates Acceptance Filter Mask Register 3 and Acceptance Filter ID Register 3 are used for acceptance filtering. 0 = Indicates Acceptance Filter Mask Register 3 and Acceptance Filter ID Register 3 are not used for acceptance filtering.
UAF2	1	rw	0x0	Use Acceptance Filter Number 2 Enables the use of acceptance filter pair 2. 1 = Indicates Acceptance Filter Mask Register 2 and Acceptance Filter ID Register 2 are used for acceptance filtering. 0 = Indicates Acceptance Filter Mask Register 2 and Acceptance Filter ID Register 2 are not used for acceptance filtering.
UAF1	0	rw	0x0	Use Acceptance Filter Number 1. Enables the use of acceptance filter pair 1. 1 = Indicates Acceptance Filter Mask Register 1 and Acceptance Filter ID Register 1 are used for acceptance filtering. 0 = Indicates Acceptance Filter Mask Register 1 and Acceptance Filter ID Register 1 are not used for acceptance filtering.

Register ([can](#)) AFMR1

Name	AFMR1
Relative Address	0x00000064
Absolute Address	can0: 0xE0008064 can1: 0xE0009064
Width	32 bits
Access Type	rw
Reset Value	x
Description	Acceptance Filter Mask Register 1

Register AFMR1 Details

Field Name	Bits	Type	Reset Value	Description
AMIDH	31:21	rw	x	Standard Message ID Mask These bits are used for masking the Identifier in a Standard Frame. 1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier. 0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.
AMSRR	20	rw	x	Substitute Remote Transmission Request Mask This bit is used for masking the RTR bit in a Standard Frame. 1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier. 0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.

Field Name	Bits	Type	Reset Value	Description
AMIDE	19	rw	x	<p>Identifier Extension Mask</p> <p>Used for masking the IDE bit in CAN frames.</p> <p>1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier.</p> <p>0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.</p> <p>If AMIDE = 1 and the AIIDE bit in the corresponding Acceptance ID register is 0, this mask is applicable to only Standard frames.</p> <p>If AMIDE = 1 and the AIIDE bit in the corresponding Acceptance ID register is 1, this mask is applicable to only extended frames.</p> <p>If AMIDE = 0 this mask is applicable to Standard frame.</p>
AMIDL	18:1	rw	x	<p>Extended Message ID Mask</p> <p>These bits are used for masking the Identifier in an Extended Frame.</p> <p>1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier.</p> <p>0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.</p>
AMRTR	0	rw	x	<p>Remote Transmission Request Mask.</p> <p>This bit is used for masking the RTR bit in an Extended Frame.</p> <p>1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier.</p> <p>0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.</p>

Register ([can](#)) AFIR1

Name	AFIR1
Relative Address	0x00000068
Absolute Address	can0: 0xE0008068 can1: 0xE0009068
Width	32 bits
Access Type	rw
Reset Value	x
Description	Acceptance Filter ID Register 1

Register AFIR1 Details

Field Name	Bits	Type	Reset Value	Description
AIIDH	31:21	rw	x	Standard Message ID Standard Identifier
AISRR	20	rw	x	Substitute Remote Transmission Request Indicates the Remote Transmission Request bit for Standard frames
AIIDE	19	rw	x	Identifier Extension Differentiates between Standard and Extended frames
AIIDL	18:1	rw	x	Extended Message ID Mask Extended Identifier
AIRTR	0	rw	x	Remote Transmission Request Mask RTR bit for Extended frames.

Register ([can](#)) AFMR2

Name	AFMR2
Relative Address	0x0000006C
Absolute Address	can0: 0xE000806C can1: 0xE000906C
Width	32 bits
Access Type	rw
Reset Value	x
Description	Acceptance Filter Mask Register 2

Register AFMR2 Details

Field Name	Bits	Type	Reset Value	Description
AMIDH	31:21	rw	x	<p>Standard Message ID Mask</p> <p>These bits are used for masking the Identifier in a Standard Frame.</p> <p>1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier.</p> <p>0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.</p>
AMSRR	20	rw	x	<p>Substitute Remote Transmission Request Mask</p> <p>This bit is used for masking the RTR bit in a Standard Frame.</p> <p>1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier.</p> <p>0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.</p>
AMIDE	19	rw	x	<p>Identifier Extension Mask</p> <p>Used for masking the IDE bit in CAN frames.</p> <p>1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier.</p> <p>0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.</p> <p>If AMIDE = 1 and the AIIDE bit in the corresponding Acceptance ID register is 0, this mask is applicable to only Standard frames.</p> <p>If AMIDE = 1 and the AIIDE bit in the corresponding Acceptance ID register is 1, this mask is applicable to only extended frames.</p> <p>If AMIDE = 0 this mask is applicable to Standard frame.</p>

Field Name	Bits	Type	Reset Value	Description
AMIDL	18:1	rw	x	Extended Message ID Mask These bits are used for masking the Identifier in an Extended Frame. 1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier. 0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.
AMRTR	0	rw	x	Remote Transmission Request Mask. This bit is used for masking the RTR bit in an Extended Frame. 1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier. 0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.

Register ([can](#)) AFIR2

Name	AFIR2
Relative Address	0x00000070
Absolute Address	can0: 0xE0008070 can1: 0xE0009070
Width	32 bits
Access Type	rw
Reset Value	x
Description	Acceptance Filter ID Register 2

Register AFIR2 Details

Field Name	Bits	Type	Reset Value	Description
AIIDH	31:21	rw	x	Standard Message ID Standard Identifier
AISRR	20	rw	x	Substitute Remote Transmission Request Indicates the Remote Transmission Request bit for Standard frames

Field Name	Bits	Type	Reset Value	Description
AIIDE	19	rw	x	Identifier Extension Differentiates between Standard and Extended frames
AIIDL	18:1	rw	x	Extended Message ID Mask Extended Identifier
AIRTR	0	rw	x	Remote Transmission Request Mask RTR bit for Extended frames.

Register ([can](#)) AFMR3

Name	AFMR3
Relative Address	0x00000074
Absolute Address	can0: 0xE0008074 can1: 0xE0009074
Width	32 bits
Access Type	rw
Reset Value	x
Description	Acceptance Filter Mask Register 3

Register AFMR3 Details

Field Name	Bits	Type	Reset Value	Description
AMIDH	31:21	rw	x	<p>Standard Message ID Mask</p> <p>These bits are used for masking the Identifier in a Standard Frame.</p> <p>1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier.</p> <p>0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.</p>
AMSRR	20	rw	x	<p>Substitute Remote Transmission Request Mask</p> <p>This bit is used for masking the RTR bit in a Standard Frame.</p> <p>1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier.</p> <p>0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.</p>
AMIDE	19	rw	x	<p>Identifier Extension Mask</p> <p>Used for masking the IDE bit in CAN frames.</p> <p>1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier.</p> <p>0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.</p> <p>If AMIDE = 1 and the AIIDE bit in the corresponding Acceptance ID register is 0, this mask is applicable to only Standard frames.</p> <p>If AMIDE = 1 and the AIIDE bit in the corresponding Acceptance ID register is 1, this mask is applicable to only extended frames.</p> <p>If AMIDE = 0 this mask is applicable to Standard frame.</p>

Field Name	Bits	Type	Reset Value	Description
AMIDL	18:1	rw	x	Extended Message ID Mask These bits are used for masking the Identifier in an Extended Frame. 1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier. 0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.
AMRTR	0	rw	x	Remote Transmission Request Mask. This bit is used for masking the RTR bit in an Extended Frame. 1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier. 0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.

Register ([can](#)) AFIR3

Name	AFIR3
Relative Address	0x00000078
Absolute Address	can0: 0xE0008078 can1: 0xE0009078
Width	32 bits
Access Type	rw
Reset Value	x
Description	Acceptance Filter ID Register 3

Register AFIR3 Details

Field Name	Bits	Type	Reset Value	Description
AIIDH	31:21	rw	x	Standard Message ID Standard Identifier
AISRR	20	rw	x	Substitute Remote Transmission Request Indicates the Remote Transmission Request bit for Standard frames

Field Name	Bits	Type	Reset Value	Description
AIIDE	19	rw	x	Identifier Extension Differentiates between Standard and Extended frames
AIIDL	18:1	rw	x	Extended Message ID Mask Extended Identifier
AIRTR	0	rw	x	Remote Transmission Request Mask RTR bit for Extended frames.

Register ([can](#)) AFMR4

Name	AFMR4
Relative Address	0x0000007C
Absolute Address	can0: 0xE000807C can1: 0xE000907C
Width	32 bits
Access Type	rw
Reset Value	x
Description	Acceptance Filter Mask Register 4

Register AFMR4 Details

Field Name	Bits	Type	Reset Value	Description
AMIDH	31:21	rw	x	<p>Standard Message ID Mask</p> <p>These bits are used for masking the Identifier in a Standard Frame.</p> <p>1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier.</p> <p>0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.</p>
AMSRR	20	rw	x	<p>Substitute Remote Transmission Request Mask</p> <p>This bit is used for masking the RTR bit in a Standard Frame.</p> <p>1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier.</p> <p>0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.</p>
AMIDE	19	rw	x	<p>Identifier Extension Mask</p> <p>Used for masking the IDE bit in CAN frames.</p> <p>1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier.</p> <p>0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.</p> <p>If AMIDE = 1 and the AIIDE bit in the corresponding Acceptance ID register is 0, this mask is applicable to only Standard frames.</p> <p>If AMIDE = 1 and the AIIDE bit in the corresponding Acceptance ID register is 1, this mask is applicable to only extended frames.</p> <p>If AMIDE = 0 this mask is applicable to Standard frame.</p>

Field Name	Bits	Type	Reset Value	Description
AMIDL	18:1	rw	x	Extended Message ID Mask These bits are used for masking the Identifier in an Extended Frame. 1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier. 0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.
AMRTR	0	rw	x	Remote Transmission Request Mask. This bit is used for masking the RTR bit in an Extended Frame. 1= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier. 0 = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.

Register ([can](#)) AFIR4

Name	AFIR4
Relative Address	0x00000080
Absolute Address	can0: 0xE0008080 can1: 0xE0009080
Width	32 bits
Access Type	rw
Reset Value	x
Description	Acceptance Filter ID Register 4

Register AFIR4 Details

Field Name	Bits	Type	Reset Value	Description
AIIDH	31:21	rw	x	Standard Message ID Standard Identifier
AISRR	20	rw	x	Substitute Remote Transmission Request Indicates the Remote Transmission Request bit for Standard frames

Field Name	Bits	Type	Reset Value	Description
AIIDE	19	rw	x	Identifier Extension Differentiates between Standard and Extended frames
AIIDL	18:1	rw	x	Extended Message ID Mask Extended Identifier
AIRTR	0	rw	x	Remote Transmission Request Mask RTR bit for Extended frames.

B.6 DDR Memory Controller (ddrc)

Module Name	DDR Memory Controller (ddrc)
Base Address	0xF8006000 ddrc
Description	DDR memory controller
Version	1.4
Doc Version	1.25
Vendor Info	Virage Logic/Synopsys

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
ddrc_ctrl	0x00000000	32	rw	0x00000200	DDRC Control
HPR_reg	0x00000008	26	rw	0x03C0780F	HPR Queue control
LPR_reg	0x0000000C	26	rw	0x03C0780F	LPR Queue control
WR_reg	0x00000010	26	rw	0x0007F80F	WR Queue control
DRAM_param_reg0	0x00000014	21	rw	0x00041016	DRAM Parameters 0
DRAM_param_reg1	0x00000018	32	rw	0x351B48D9	DRAM Parameters 1
DRAM_param_reg2	0x0000001C	32	rw	0x83015904	DRAM Parameters 2
DRAM_param_reg3	0x00000020	32	mixed	0x250882D0	DRAM Parameters 3
DRAM_param_reg4	0x00000024	28	mixed	0x0000003C	DRAM Parameters 4
DRAM_init_param	0x00000028	14	rw	0x00002007	DRAM Initialization Parameters
DRAM_EMR_reg	0x0000002C	32	rw	0x00000008	DRAM EMR2, EMR3 access
DRAM_EMR_MR_reg	0x00000030	32	rw	0x00000940	DRAM EMR, MR access
DRAM_burst8_rdwr	0x00000034	29	mixed	0x00020034	DRAM Burst 8 read/write
DRAM_disable_DQ	0x00000038	13	mixed	0x00000000	DRAM Disable DQ
DRAM_addr_map_bank	0x0000003C	20	rw	0x00000F77	Row/Column address bits
DRAM_addr_map_col	0x00000040	32	rw	0xFFF00000	Column address bits
DRAM_addr_map_row	0x00000044	28	rw	0x0FF55555	Select DRAM row address bits
DRAM_ODT_reg	0x00000048	30	rw	0x00000249	DRAM ODT control
phy_dbg_reg	0x0000004C	20	ro	0x00000000	PHY debug
phy_cmd_timeout_rddata_cpt	0x00000050	32	mixed	0x00010200	PHY command time out and read data capture FIFO

Register Name	Address	Width	Type	Reset Value	Description
mode_sts_reg	0x00000054	21	ro	0x00000000	Controller operation mode status
DLL_calib	0x00000058	17	rw	0x00000101	DLL calibration
ODT_delay_hold	0x0000005C	16	rw	0x00000023	ODT delay and ODT hold
ctrl_reg1	0x00000060	13	mixed	0x0000003E	Controller 1
ctrl_reg2	0x00000064	18	mixed	0x00020000	Controller 2
ctrl_reg3	0x00000068	26	rw	0x00284027	Controller 3
ctrl_reg4	0x0000006C	16	rw	0x00001610	Controller 4
ctrl_reg5	0x00000078	32	mixed	0x00455111	Controller register 5
ctrl_reg6	0x0000007C	32	mixed	0x00032222	Controller register 6
CHE_REFRESH_TIMER01	0x000000A0	24	rw	0x00008000	CHE_REFRESH_TIMER01
CHE_T_ZQ	0x000000A4	32	rw	0x10300802	ZQ parameters
CHE_T_ZQ_Short_Inteval_Reg	0x000000A8	28	rw	0x0020003A	Misc parameters
deep_pwrdown_reg	0x000000AC	9	rw	0x00000000	Deep powerdown (LPDDR2)
reg_2c	0x000000B0	29	mixed	0x00000000	Training control
reg_2d	0x000000B4	11	rw	0x00000200	Misc Debug
dfi_timing	0x000000B8	25	rw	0x00200067	DFI timing
CHE_ECC_CONTROL_REG_OFFSET	0x000000C4	2	rw	0x00000000	ECC error clear
CHE_CORR_ECC_LOG_REG_OFFSET	0x000000C8	8	mixed	0x00000000	ECC error correction
CHE_CORR_ECC_ADDR_REG_OFFSET	0x000000CC	31	ro	0x00000000	ECC error correction address log
CHE_CORR_ECC_DATA_31_0_REG_OFFSET	0x000000D0	32	ro	0x00000000	ECC error correction data log low
CHE_CORR_ECC_DATA_63_32_REG_OFFSET	0x000000D4	32	ro	0x00000000	ECC error correction data log mid
CHE_CORR_ECC_DATA_71_64_REG_OFFSET	0x000000D8	8	ro	0x00000000	ECC error correction data log high
CHE_UNCORR_ECC_LOG_REG_OFFSET	0x000000DC	1	clon wr	0x00000000	ECC unrecoverable error status
CHE_UNCORR_ECC_ADDR_REG_OFFSET	0x000000E0	31	ro	0x00000000	ECC unrecoverable error address

Register Name	Address	Width	Type	Reset Value	Description
CHE_UNCORR_ECC_DATA_31_0_REG_OFFSET	0x000000E4	32	ro	0x00000000	ECC unrecoverable error data low
CHE_UNCORR_ECC_DATA_63_32_REG_OFFSET	0x000000E8	32	ro	0x00000000	ECC unrecoverable error data middle
CHE_UNCORR_ECC_DATA_71_64_REG_OFFSET	0x000000EC	8	ro	0x00000000	ECC unrecoverable error data high
CHE_ECC_STATS_REG_OFFSET	0x000000F0	16	clon wr	0x00000000	ECC error count
ECC_scrub	0x000000F4	4	rw	0x00000008	ECC mode/scrub
CHE_ECC_CORR_BIT_MASK_31_0_REG_OFFSET	0x000000F8	32	ro	0x00000000	ECC data mask low
CHE_ECC_CORR_BIT_MASK_63_32_REG_OFFSET	0x000000FC	32	ro	0x00000000	ECC data mask high
phy_rcvr_enable	0x00000114	8	rw	0x00000000	Phy receiver enable register
PHY_Config0	0x00000118	31	rw	0x40000001	PHY configuration register for data slice 0.
PHY_Config1	0x0000011C	31	rw	0x40000001	PHY configuration register for data slice 1.
PHY_Config2	0x00000120	31	rw	0x40000001	PHY configuration register for data slice 2.
PHY_Config3	0x00000124	31	rw	0x40000001	PHY configuration register for data slice 3.
phy_init_ratio0	0x0000012C	20	rw	0x00000000	PHY init ratio register for data slice 0.
phy_init_ratio1	0x00000130	20	rw	0x00000000	PHY init ratio register for data slice 1.
phy_init_ratio2	0x00000134	20	rw	0x00000000	PHY init ratio register for data slice 2.
phy_init_ratio3	0x00000138	20	rw	0x00000000	PHY init ratio register for data slice 3.
phy_rd_dqs_cfg0	0x00000140	20	rw	0x00000040	PHY read DQS configuration register for data slice 0.
phy_rd_dqs_cfg1	0x00000144	20	rw	0x00000040	PHY read DQS configuration register for data slice 1.
phy_rd_dqs_cfg2	0x00000148	20	rw	0x00000040	PHY read DQS configuration register for data slice 2.

Register Name	Address	Width	Type	Reset Value	Description
phy_rd_dqs_cfg3	0x0000014C	20	rw	0x00000040	PHY read DQS configuration register for data slice 3.
phy_wr_dqs_cfg0	0x00000154	20	rw	0x00000000	PHY write DQS configuration register for data slice 0.
phy_wr_dqs_cfg1	0x00000158	20	rw	0x00000000	PHY write DQS configuration register for data slice 1.
phy_wr_dqs_cfg2	0x0000015C	20	rw	0x00000000	PHY write DQS configuration register for data slice 2.
phy_wr_dqs_cfg3	0x00000160	20	rw	0x00000000	PHY write DQS configuration register for data slice 3.
phy_we_cfg0	0x00000168	21	rw	0x00000040	PHY FIFO write enable configuration for data slice 0.
phy_we_cfg1	0x0000016C	21	rw	0x00000040	PHY FIFO write enable configuration for data slice 1.
phy_we_cfg2	0x00000170	21	rw	0x00000040	PHY FIFO write enable configuration for data slice 2.
phy_we_cfg3	0x00000174	21	rw	0x00000040	PHY FIFO write enable configuration for data slice 3.
wr_data_slv0	0x0000017C	20	rw	0x00000080	PHY write data slave ratio config for data slice 0.
wr_data_slv1	0x00000180	20	rw	0x00000080	PHY write data slave ratio config for data slice 1.
wr_data_slv2	0x00000184	20	rw	0x00000080	PHY write data slave ratio config for data slice 2.
wr_data_slv3	0x00000188	20	rw	0x00000080	PHY write data slave ratio config for data slice 3.
reg_64	0x00000190	32	rw	0x10020000	Training control 2
reg_65	0x00000194	20	rw	0x00000000	Training control 3
reg69_6a0	0x000001A4	29	ro	0x000F0000	Training results for data slice 0.
reg69_6a1	0x000001A8	29	ro	0x000F0000	Training results for data slice 1.
reg6c_6d2	0x000001B0	29	ro	0x000F0000	Training results for data slice 2.
reg6c_6d3	0x000001B4	29	ro	0x000F0000	Training results for data slice 3.
reg6e_710	0x000001B8	30	ro	x	Training results (2) for data slice 0.
reg6e_711	0x000001BC	30	ro	x	Training results (2) for data slice 1.
reg6e_712	0x000001C0	30	ro	x	Training results (2) for data slice 2.
reg6e_713	0x000001C4	30	ro	x	Training results (2) for data slice 3.

Register Name	Address	Width	Type	Reset Value	Description
phy_dll_sts0	0x000001CC	27	ro	0x00000000	Slave DLL results for data slice 0.
phy_dll_sts1	0x000001D0	27	ro	0x00000000	Slave DLL results for data slice 1.
phy_dll_sts2	0x000001D4	27	ro	0x00000000	Slave DLL results for data slice 2.
phy_dll_sts3	0x000001D8	27	ro	0x00000000	Slave DLL results for data slice 3.
dll_lock_sts	0x000001E0	24	ro	0x00000000	DLL Lock Status, read
phy_ctrl_sts	0x000001E4	30	ro	x	PHY Control status, read
phy_ctrl_sts_reg2	0x000001E8	27	ro	0x00000000	PHY Control status (2), read
axi_id	0x00000200	26	ro	0x00153042	ID and revision information
page_mask	0x00000204	32	rw	0x00000000	Page mask
axi_priority_wr_port0	0x00000208	20	mixed	0x000803FF	AXI Priority control for write port 0.
axi_priority_wr_port1	0x0000020C	20	mixed	0x000803FF	AXI Priority control for write port 1.
axi_priority_wr_port2	0x00000210	20	mixed	0x000803FF	AXI Priority control for write port 2.
axi_priority_wr_port3	0x00000214	20	mixed	0x000803FF	AXI Priority control for write port 3.
axi_priority_rd_port0	0x00000218	20	mixed	0x000003FF	AXI Priority control for read port 0.
axi_priority_rd_port1	0x0000021C	20	mixed	0x000003FF	AXI Priority control for read port 1.
axi_priority_rd_port2	0x00000220	20	mixed	0x000003FF	AXI Priority control for read port 2.
axi_priority_rd_port3	0x00000224	20	mixed	0x000003FF	AXI Priority control for read port 3.
trusted_mem_cfg	0x00000290	16	rw	0x00000000	Trusted Memory configuration
excl_access_cfg0	0x00000294	18	rw	0x00000000	Exclusive access configuration for port 0.
excl_access_cfg1	0x00000298	18	rw	0x00000000	Exclusive access configuration for port 1.
excl_access_cfg2	0x0000029C	18	rw	0x00000000	Exclusive access configuration for port 2.
excl_access_cfg3	0x000002A0	18	rw	0x00000000	Exclusive access configuration for port 3.
mode_reg_read	0x000002A4	32	ro	0x00000000	Mode register read data
lpddr_ctrl0	0x000002A8	12	rw	0x00000000	LPDDR2 Control 0

Register Name	Address	Width	Type	Reset Value	Description
lpddr_ctrl1	0x000002AC	32	rw	0x00000000	LPDDR2 Control 1
lpddr_ctrl2	0x000002B0	22	rw	0x003C0015	LPDDR2 Control 2
lpddr_ctrl3	0x000002B4	18	rw	0x00000601	LPDDR2 Control 3
phy_wr_lvl_fsm	0x000002B8	15	ro	0x00004444	PHY write leveling state machine status, read
phy_rd_lvl_fsm	0x000002BC	16	ro	0x00008888	PHY read leveling state machine status
phy_gate_lvl_fsm	0x000002C0	15	ro	0x00004444	PHY gate leveling state machine status

Register ([ddrc](#)) ddrc_ctrl

Name	ddrc_ctrl
Relative Address	0x00000000
Absolute Address	0xF8006000
Width	32 bits
Access Type	rw
Reset Value	0x00000200
Description	DDRC Control

Register ddrc_ctrl Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:17	rw	0x0	reserved
reg_ddrc_dis_auto_refresh	16	rw	0x0	Disable auto-refresh. 0: do not disable auto-refresh. 1: disable auto-refresh. Dynamic Bit Field. Note: When this transitions from 0 to 1, any pending refreshes will be immediately scheduled by the controller.
reg_ddrc_dis_act_bypass	15	rw	0x0	Only present in designs supporting activate bypass. For Debug only. 0: Do not disable bypass path for high priority read activates. 1: disable bypass path for high priority read activates.

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_dis_rd_bypass	14	rw	0x0	Only present in designs supporting read bypass. For Debug only. 0: Do not disable bypass path for high priority read page hits. 1: disable bypass path for high priority read page hits.
reg_ddrc_rdwr_idle_gap	13:7	rw	0x4	When the preferred transaction store is empty for this many clock cycles, switch to the alternate transaction store if it is non-empty. The read transaction store (both high and low priority) is the default preferred transaction store and the write transaction store is the alternate store. When 'Prefer write over read' is set this is reversed.
reg_ddrc_burst8_refresh	6:4	rw	0x0	Refresh timeout. Programmed value plus one will be the number of refresh timeouts that will be allowed to accumulate before traffic is blocked and the refreshes are forced to execute. Closing pages to perform a refresh is a one-time penalty that must be paid for each group of refreshes; therefore, performing refreshes in a burst reduces the per-refresh penalty of these page closings. Higher numbers for burst_of_N_refresh slightly increases utilization; lower numbers decreases the worst-case latency associated with refreshes. 0: single refresh 1: burst-of-2 ... 7: burst-of-8 refresh
reg_ddrc_data_bus_width	3:2	rw	0x0	DDR bus width control 00: 32-bit 01: 16-bit 1x: reserved

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_powerdown_en	1	rw	0x0	Controller power down control. Update during normal operation. Enable the controller to powerdown after it becomes idle. Dynamic Bit Field. 0: disable 1: enable
reg_ddrc_soft_rstb	0	rw	0x0	Active low soft reset. Update during normal operation. 0: Resets the controller 1: Takes the controller out of reset. Dynamic Bit Field. Note: Software changes DRAM controller register values only when the controller is in the reset state, except for bit fields that can be dynamically updated.

Register ([ddrc](#)) HPR_reg

Name	HPR_reg
Relative Address	0x00000008
Absolute Address	0xF8006008
Width	26 bits
Access Type	rw
Reset Value	0x03C0780F
Description	HPR Queue control

Register HPR_reg Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_hpr_xact_run_length	25:22	rw	0xF	Number of transactions that will be serviced once the HPR queue goes critical is the smaller of this number and the number of transactions available.
reg_ddrc_hpr_max_starve_x32	21:11	rw	0xF	Number of clocks that the HPR queue can be starved before it goes critical. Unit: 32 clocks
reg_ddrc_hpr_min_non_critical_x32	10:0	rw	0xF	Number of counts that the HPR queue is guaranteed to be non-critical (1 count = 32 DDR clocks).

Register ([ddrc](#)) LPR_reg

Name	LPR_reg
------	---------

Relative Address	0x0000000C
Absolute Address	0xF800600C
Width	26 bits
Access Type	rw
Reset Value	0x03C0780F
Description	LPR Queue control

Register LPR_reg Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_lpr_xact_run_length	25:22	rw	0xF	Number of transactions that will be serviced once the LPR queue goes critical is the smaller of this number and the number of transactions available
reg_ddrc_lpr_max_starve_x32	21:11	rw	0xF	Number of clocks that the LPR queue can be starved before it goes critical. Unit: 32 clocks
reg_ddrc_lpr_min_non_critical_x32	10:0	rw	0xF	Number of clocks that the LPR queue is guaranteed to be non-critical. Unit: 32 clocks

Register ([ddrc](#)) WR_reg

Name	WR_reg
Relative Address	0x00000010
Absolute Address	0xF8006010
Width	26 bits
Access Type	rw
Reset Value	0x0007F80F
Description	WR Queue control

Register WR_reg Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_w_max_starve_x32	25:15	rw	0xF	Number of clocks that the Write queue can be starved before it goes critical. Unit: 32 clocks. FOR PERFORMANCE ONLY.
reg_ddrc_w_xact_run_length	14:11	rw	0xF	Number of transactions that will be serviced once the WR queue goes critical is the smaller of this number and the number of transactions available
reg_ddrc_w_min_non_critical_x32	10:0	rw	0xF	Number of clock cycles that the WR queue is guaranteed to be non-critical.

Register ([ddrc](#)) DRAM_param_reg0

Name	DRAM_param_reg0
Relative Address	0x00000014
Absolute Address	0xF8006014
Width	21 bits
Access Type	rw
Reset Value	0x00041016
Description	DRAM Parameters 0

Register DRAM_param_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_post_selfref_gap_x32	20:14	rw	0x10	Minimum time to wait after coming out of self refresh before doing anything. This must be bigger than all the constraints that exist. (spec: Maximum of tXSNR and tXSRD and tXSDLL which is 512 clocks). Unit: in multiples of 32 clocks. DRAM Related
reg_ddrc_t_rfc_min	13:6	rw	0x40	tRFC(min) - Minimum time from refresh to refresh or activate (spec: 75nS to 195nS). DRAM Related. Default value is set for DDR3. Dynamic Bit Field.
reg_ddrc_t_rc	5:0	rw	0x16	tRC - Min time between activates to same bank (spec: 65 ns for DDR2-400 and smaller for faster parts). DRAM Related. Default value is set for DDR3.

Register ([ddrc](#)) DRAM_param_reg1

Name	DRAM_param_reg1
Relative Address	0x00000018
Absolute Address	0xF8006018
Width	32 bits
Access Type	rw
Reset Value	0x351B48D9
Description	DRAM Parameters 1

Register DRAM_param_reg1 Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_t_cke	31:28	rw	0x3	Minimum number of cycles of CKE HIGH/LOW during power down and self refresh. DDR2 and DDR3: Set this to tCKE value. LPDDR2: Set this to the larger of tCKE or tCKESR. Unit: clocks.
reg_ddrc_t_ras_min	26:22	rw	0x14	tRAS(min) - Minimum time between activate and precharge to the same bank (spec is 45 ns). Unit: clocks DRAM related. Default value is set for DDR3.
reg_ddrc_t_ras_max	21:16	rw	0x1B	tRAS(max) - Maximum time between activate and precharge to same bank. Maximum time that a page can be kept open (spec is 70 us). If this is zero. The page is closed after each transaction. Unit: Multiples of 1024 clocks DRAM related.
reg_ddrc_t_faw	15:10	rw	0x12	tFAW - At most 4 banks must be activated in a rolling window of tFAW cycles. Unit: clocks. DRAM Related.
reg_ddrc_powerdown_to_x32	9:5	rw	0x6	After this many clocks of NOP or DESELECT the controller will put the DRAM into power down. This must be enabled in the Master Control Register. Unit: Multiples of 32 clocks.
reg_ddrc_wr2pre	4:0	rw	0x19	Minimum time between write and precharge to same bank DDR and DDR3: $WL + BL/2 + tWR$ LPDDR2: $WL + BL/2 + tWR + 1$ Unit: Clocks where, WL: write latency. BL: burst length. This must match the value programmed in the BL bit of the mode register to the DRAM. BST is not supported at present. tWR: write recovery time. This comes directly from the DRAM specs.

Register ([ddrc](#)) DRAM_param_reg2

Name	DRAM_param_reg2
Relative Address	0x0000001C
Absolute Address	0xF800601C
Width	32 bits
Access Type	rw

Reset Value 0x83015904
Description DRAM Parameters 2

Register DRAM_param_reg2 Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_t_rcd	31:28	rw	0x8	tRCD - AL Minimum time from activate to read or write command to same bank Min value for this is 1. AL = Additive Latency. DRAM Related.
reg_ddrc_rd2pre	27:23	rw	0x6	Minimum time from read to precharge of same bank DDR2: $AL + BL/2 + \max(tRTP, 2) - 2$ DDR3: $AL + \max(tRTP, 4)$ LPDDR2: $BL/2 + tRTP - 1$ AL: Additive Latency; BL: DRAM Burst Length; tRTP: value from spec. DRAM related.
reg_ddrc_pad_pd	22:20	rw	0x0	If pads have a power-saving mode, this is the greater of the time for the pads to enter power down or the time for the pads to exit power down. Used only in non-DFI designs. Unit: clocks.
reg_ddrc_t_xp	19:15	rw	0x2	tXP: Minimum time after power down exit to any operation. DRAM related.
reg_ddrc_wr2rd	14:10	rw	0x16	Minimum time from write command to read command. Includes time for bus turnaround and recovery times and all per-bank, per-rank, and global constraints. DDR2 and DDR3: $WL + tWTR + BL/2$ LPDDR2: $WL + tWTR + BL/2 + 1$ Unit: clocks. Where, WL: Write latency, BL: burst length. This should match the value. Programmed in the BL bit of the mode register to the DRAM. tWTR: internal WRITE to READ command delay. This comes directly from the DRAM specs.

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_rd2wr	9:5	rw	0x8	Minimum time from read command to write command. Include time for bus turnaround and all per-bank, per-rank, and global constraints. DDR2 and DDR3: $RL + BL/2 + 2 - WL$ LPDDR2: $RL + BL/2 + RU (tDQSK_{max} / tCK) + 1 - WL$ Write Pre-amble and DQ/DQS jitter timer is included in the above equation. DRAM RELATED.
reg_ddrc_write_latency	4:0	rw	0x4	Time from write command to write data on DDRC to PHY Interface. (PHY adds an extra flop delay on the write data path; hence this value is one less than the write latency of the DRAM device itself). DDR2 and DDR3: $WL - 1$ LPDDR2: WL Where WL : Write Latency of DRAM DRAM related. In non-LPDDR mode, the minimum DRAM Write Latency (DDR2) supported is 3. In LPDDR mode, the required DRAM Write Latency of 1 is supported. Since write latency (CWL) min is 3, and DDR2 CWL is CL-1, the min (DDR2) CL supported is 4

Register ([ddrc](#)) DRAM_param_reg3

Name	DRAM_param_reg3
Relative Address	0x00000020
Absolute Address	0xF8006020
Width	32 bits
Access Type	mixed
Reset Value	0x250882D0
Description	DRAM Parameters 3

Register DRAM_param_reg3 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	rw	0x0	Reserved. Do not modify.
reg_ddrc_dis_pad_pd	30	rw	0x0	1: disable the pad power down feature 0: Enable the pad power down feature.

Field Name	Bits	Type	Reset Value	Description
reg_phy_mode_ddr1_dr2	29	rw	0x1	unused
reg_ddrc_read_latency	28:24	rw	0x5	Non-LPDDR2: not used. DDR2 and DDR3: Set to Read Latency, RL. Time from Read command to Read data on DRAM interface. It is used to calculate when DRAM clock may be stopped. Unit: DDR clock.
reg_ddrc_en_dfi_dram_clk_disable	23	rw	0x0	Enables the assertion of ddrdc_dfi_dram_clk_disable. In DDR2/DDR3, only asserted in Self Refresh. In mDDR/LPDDR2, can be asserted in following: - during normal operation (Clock Stop), - in Power Down - in Self Refresh - In Deep Power Down
reg_ddrc_mobile	22	rw	0x0	0: DDR2 or DDR3 device. 1: LPDDR2 device.
reserved	21	rw	0x0	Reserved. Do not modify.
reg_ddrc_refresh_to_x32	20:16	rw	0x8	If the refresh timer (tRFC_nom, as known as tREFI) has expired at least once, but it has not expired burst_of_N_refresh times yet, then a 'speculative refresh' may be performed. A speculative refresh is a refresh performed at a time when refresh would be useful, but before it is absolutely required. When the DRAM bus is idle for a period of time determined by this refresh idle timeout and the refresh timer has expired at least once since the last refresh, then a 'speculative refresh' will be performed. Speculative refreshes will continue successively until there are no refreshes pending or until new reads or writes are issued to the controller. Dynamic Bit Field.
reg_ddrc_t_rp	15:12	rw	0x8	tRP - Minimum time from precharge to activate of same bank. DRAM RELATED
reg_ddrc_refresh_margin	11:8	rw	0x2	Issue critical refresh or page close this many cycles before the critical refresh or page timer expires. It is recommended that this not be changed from the default value.
reg_ddrc_t_rrd	7:5	rw	0x6	tRRD - Minimum time between activates from bank A to bank B. (spec: 10ns or less) DRAM RELATED

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_t_ccd	4:2	rw	0x4	tCCD - Minimum time between two reads or two writes (from bank a to bank b) is this value + 1. DRAM related.
reserved	1:0	ro	0x0	Reserved

Register ([ddrc](#)) DRAM_param_reg4

Name	DRAM_param_reg4
Relative Address	0x00000024
Absolute Address	0xF8006024
Width	28 bits
Access Type	mixed
Reset Value	0x0000003C
Description	DRAM Parameters 4

Register DRAM_param_reg4 Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_mr_rdata_val id	27	clonr d	0x0	This bit indicates whether the Mode Register Read Data present at address 0xA9 is valid or not. This bit is 0 by default. This bit will be cleared (0), whenever a Mode Register Read command is issued. This bit will be set to 1, when the Mode Register Read Data is written to register 0xA9.
reg_ddrc_mr_type	26	rw	0x0	Indicates whether the Mode register operation is read or write 0: write 1: read
ddrc_reg_mr_wr_busy	25	ro	0x0	Core must initiate a MR write / read operation only if this signal is low. This signal goes high in the clock after the controller accepts the write / read request. It goes low when (i) MR write command has been issued to the DRAM (ii) MR Read data has been returned to Controller. Any MR write / read command that is received when 'ddrc_reg_mr_wr_busy' is high is not accepted. 0: Indicates that the core can initiate a mode register write / read operation. 1: Indicates that mode register write / read operation is in progress.

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_mr_data	24:9	rw	0x0	DDR2 and DDR3: Mode register write data. LPDDR2: The 16 bits are interpreted for reads and writes: Reads: MR Addr[7:0], Don't Care[7:0]. Writes: MR Addr[7:0], MR Data[7:0].
reg_ddrc_mr_addr	8:7	rw	0x0	DDR2 and DDR3: Mode register address. LPDDR2: not used. 00: MR0 01: MR1 10: MR2 11: MR3
reg_ddrc_mr_wr	6	wo	0x0	A low to high signal on this signal will do a mode register write or read. Controller will accept this command, if this signal is detected high and "ddrc_reg_mr_wr_busy" is detected low.
reserved	5:2	rw	0xF	Reserved. Do not modify.
reg_ddrc_prefer_write	1	rw	0x0	1: Bank selector prefers writes over reads
reg_ddrc_en_2t_timing_mode	0	rw	0x0	1: DDRC will use 2T timing 0: DDRC will use 1T timing

Register ([ddrc](#)) DRAM_init_param

Name	DRAM_init_param
Relative Address	0x00000028
Absolute Address	0xF8006028
Width	14 bits
Access Type	rw
Reset Value	0x00002007
Description	DRAM Initialization Parameters

Register DRAM_init_param Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_t_mrd	13:11	rw	0x4	tMRD - Cycles between Load Mode commands. DRAM related. Default value is set for DDR3.

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_pre_ocrd_x32	10:7	rw	0x0	Wait period before driving the 'OCD Complete' command to DRAM. Units are in counts of a global timer that pulses every 32 clock cycles. There is no known spec requirement for this. It may be set to zero.
reg_ddrc_final_wait_x32	6:0	rw	0x7	Cycles to wait after completing the DRAM init sequence before starting the dynamic scheduler. Units are in counts of a global timer that pulses every 32 clock cycles. Default value is set for DDR3.

Register ([ddrc](#)) DRAM_EMR_reg

Name	DRAM_EMR_reg
Relative Address	0x0000002C
Absolute Address	0xF800602C
Width	32 bits
Access Type	rw
Reset Value	0x00000008
Description	DRAM EMR2, EMR3 access

Register DRAM_EMR_reg Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_emr3	31:16	rw	0x0	DDR2 and DDR3: Value written into the DRAM EMR3 register. LPDDR2: not used.
reg_ddrc_emr2	15:0	rw	0x8	DDR2 and DDR3: Value written into the DRAM EMR2 register. LPDDR2: Value written into the DRAM MR3 register.

Register ([ddrc](#)) DRAM_EMR_MR_reg

Name	DRAM_EMR_MR_reg
Relative Address	0x00000030
Absolute Address	0xF8006030
Width	32 bits
Access Type	rw
Reset Value	0x00000940

Description DRAM EMR, MR access

Register DRAM_EMR_MR_reg Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_emr	31:16	rw	0x0	DDR2 and DDR3: Value written into the DRAM EMR registers. Bits [9:7] are for OCD and the setting in this register is ignored. The controller sets those bits appropriately. LPDDR2: Value written into the DRAM MR2 register.
reg_ddrc_mr	15:0	rw	0x940	DDR2 and DDR3: Value written into the DRAM Mode register. Bit 8 is for DLL and the setting here is ignored. The controller sets appropriately. LPDDR2: Value written into the DRAM MR1 register

Register ([ddrc](#)) DRAM_burst8_rdwr

Name DRAM_burst8_rdwr
Relative Address 0x00000034
Absolute Address 0xF8006034
Width 29 bits
Access Type mixed
Reset Value 0x00020034
Description DRAM Burst 8 read/write

Register DRAM_burst8_rdwr Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_burstchop	28	rw	0x0	Feature not supported. When 1, Controller is out in burstchop mode.
reserved	27:26	ro	0x0	Reserved
reg_ddrc_post_cke_x10 24	25:16	rw	0x2	Clock cycles to wait after driving CKE high to start the DRAM initialization sequence. Units: 1024 clocks. DDR2 typically require a 400 ns delay, requiring this value to be programmed to 2 at all clock speeds. LPDDR2 - Typically require this to be programmed for a delay of 200 us.
reserved	15:14	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_pre_cke_x1024	13:4	rw	0x3	<p>Clock cycles to wait after a DDR software reset before driving CKE high to start the DRAM initialization sequence.</p> <p>Units: 1024 clock cycles.</p> <p>DDR2 Specifications typically require this to be programmed for a delay of ≥ 200 μS.</p> <p>LPDDR2 - tINIT0 of 20 mS (max) + tINIT1 of 100 nS (min)</p>
reg_ddrc_burst_rdwr	3:0	rw	0x4	<p>Controls the burst size used to access the DRAM. This must match the BL mode register setting in the DRAM.</p> <p>0010: Burst length of 4</p> <p>0100: Burst length of 8</p> <p>1000: Burst length of 16 (LPDDR2 with ___-bit data)</p> <p>All other values are reserved</p>

Register ([ddrc](#)) DRAM_disable_DQ

Name	DRAM_disable_DQ
Relative Address	0x00000038
Absolute Address	0xF8006038
Width	13 bits
Access Type	mixed
Reset Value	0x00000000
Description	DRAM Disable DQ

Register DRAM_disable_DQ Details

Field Name	Bits	Type	Reset Value	Description
reserved	12:9	rw	0x0	Reserved. Do not modify.
reserved	8	rw	0x0	Reserved. Do not modify.
reserved	7	rw	0x0	Reserved. Do not modify.
reserved	6	rw	0x0	Reserved. Do not modify.
reserved	5:2	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_dis_dq	1	rw	0x0	When 1, DDRC will not de-queue any transactions from the CAM. Bypass will also be disabled. All transactions will be queued in the CAM. This is for debug only; no reads or writes are issued to DRAM as long as this is asserted. Dynamic Bit Field.
reg_ddrc_force_low_pri_n	0	rw	0x0	Read Transaction Priority disable. 0: read transactions forced to low priority (turns off Bypass). 1: HPR reads allowed if enabled in the AXI priority read registers.

Register ([ddrc](#)) DRAM_addr_map_bank

Name	DRAM_addr_map_bank
Relative Address	0x0000003C
Absolute Address	0xF800603C
Width	20 bits
Access Type	rw
Reset Value	0x00000F77
Description	Row/Column address bits

Register DRAM_addr_map_bank Details

Note: address bits are relative to a byte address. For example, the value 0x777 in bits[11:0] selects byte address bits [14:12] as bank address bits.

Selects the address bits used as DRAM bank address bits

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_addrmap_col_b6	19:16	rw	0x0	Full bus width mode: Selects the address bits used as column address bits 7. Half bus width mode: Selects the address bits used as column address bits 8. Valid range is 0-7. Internal Base 9. The selected address bit for each of the column address bits is determined by adding the Internal Base to the value of this field. Internal base: 9
reg_ddrc_addrmap_col_b5	15:12	rw	0x0	Full bus width mode: Selects the address bits used as column address bits 6. Half bus width mode: Selects the address bits used as column address bits 7. Valid range is 0-7. Internal Base 8. The selected address bit for each of the column address bits is determined by adding the Internal Base to the value of this field. Internal base: 9
reg_ddrc_addrmap_bank_b2	11:8	rw	0xF	Selects the AXI address bit used as bank address bit 2. Valid range 0 to 14, and 15. Internal Base: 7. The selected address bit is determined by adding the Internal Base to the value of this field. If set to 15, bank address bit 2 is set to 0.
reg_ddrc_addrmap_bank_b1	7:4	rw	0x7	Selects the address bits used as bank address bit 1. Valid Range: 0 to 14; Internal Base: 6. The selected address bit for each of the bank address bits is determined by adding the Internal Base to the value of this field.
reg_ddrc_addrmap_bank_b0	3:0	rw	0x7	Selects the address bits used as bank address bit 0. Valid Range: 0 to 14. Internal Base: 5. The selected address bit for each of the bank address bits is determined by adding the Internal Base to the value of this field.

Register ([ddrc](#)) DRAM_addr_map_col

Name	DRAM_addr_map_col
Relative Address	0x00000040
Absolute Address	0xF8006040
Width	32 bits
Access Type	rw
Reset Value	0xFFF00000
Description	Column address bits

Register DRAM_addr_map_col Details

Selects the address bits used as DRAM column address bits

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_addrmap_col_b11	31:28	rw	0xF	<p>Full bus width mode: Selects the address bit used as column address bit 13. (Column address bit 12 in LPDDR2 mode) Half bus width mode: Unused. To make it unused, this should be set to 15. (Column address bit 13 in LPDDR2 mode)</p> <p>Valid Range: 0 to 7, and 15.</p> <p>Internal Base: 14.</p> <p>The selected address bit is determined by adding the Internal Base to the value of this field. If set to 15, this column address bit is set to 0. Note: Per JEDEC DDR2 spec, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10. In LPDDR2, there is a dedicated bit for auto-precharge in the CA bus, and hence column bit 10 is used.</p>
reg_ddrc_addrmap_col_b10	27:24	rw	0xF	<p>Full bus width mode: Selects the address bit used as column address bit 12. (Column address bit 11 in LPDDR2 mode) Half bus width mode: Selects the address bit used as column address bit 13. (Column address bit 12 in LPDDR2 mode) Valid Range: 0 to 7, and 15.</p> <p>Internal Base: 13</p> <p>The selected address bit is determined by adding the Internal Base to the value of this field. If set to 15, this column address bit is set to 0. Note: Per JEDEC DDR2 spec, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10. In LPDDR2, there is a dedicated bit for auto-precharge in the CA bus, and hence column bit 10 is used.</p>
reg_ddrc_addrmap_col_b9	23:20	rw	0xF	<p>Full bus width mode: Selects the address bit used as column address bit 11. (Column address bit 10 in LPDDR2 mode)</p> <p>Half bus width mode: Selects the address bit used as column address bit 12. (Column address bit 11 in LPDDR2 mode)</p> <p>Valid Range: 0 to 7, and 15</p> <p>Internal Base: 12</p> <p>The selected address bit is determined by adding the Internal Base to the value of this field. If set to 15, this column address bit is set to 0.</p> <p>Note: Per JEDEC DDR2 spec, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10. In LPDDR2, there is a dedicated bit for auto-precharge in the CA bus, and hence column bit 10 is used.</p>

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_addrmap_col_b8	19:16	rw	0x0	<p>Full bus width mode: Selects the address bit used as column address bit 9.</p> <p>Half bus width mode: Selects the address bit used as column address bit 11. (Column address bit 10 in LPDDR2 mode)</p> <p>Valid Range: 0 to 7, and 15</p> <p>Internal Base: 11</p> <p>The selected address bit is determined by adding the Internal Base to the value of this field. If set to 15, this column address bit is set to 0.</p> <p>Note: Per JEDEC spec, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10. In LPDDR2, there is a dedicated bit for auto-precharge in the CA bus, and hence column bit 10 is used.</p>
reg_ddrc_addrmap_col_b7	15:12	rw	0x0	<p>Full bus width mode: Selects the address bit used as column address bit 8.</p> <p>Half bus width mode: Selects the address bit used as column address bit 9.</p> <p>Valid Range: 0 to 7, and 15.</p> <p>Internal Base: 10.</p> <p>The selected address bit is determined by adding the Internal Base to the value of this field. If set to 15, this column address bit is set to 0.</p> <p>Note: Per JEDEC spec, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10. In LPDDR2, there is a dedicated bit for auto-precharge in the CA bus, and hence column bit 10 is used.</p>
reg_ddrc_addrmap_col_b4	11:8	rw	0x0	<p>Full bus width mode: Selects the address bit used as column address bit 5.</p> <p>Half bus width mode: Selects the address bit used as column address bits 6. Valid Range: 0 to 7.</p> <p>Internal Base: 7.</p> <p>The selected address bit for each of the column address bits is determined by adding the Internal Base to the value of this field.</p>

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_addrmap_col_b3	7:4	rw	0x0	Full bus width mode: Selects the address bit used as column address bit 4. Half bus width mode: Selects the address bit used as column address bit 5. Valid Range: 0 to 7 Internal Base: 6 The selected address bit is determined by adding the Internal Base to the value of this field.
reg_ddrc_addrmap_col_b2	3:0	rw	0x0	Full bus width mode: Selects the address bit used as column address bit 3. Half bus width mode: Selects the address bit used as column address bit 4. Valid Range: 0 to 7. Internal Base: 5 The selected address bit is determined by adding the Internal Base to the value of this field.

Register ([ddrc](#)) DRAM_addr_map_row

Name	DRAM_addr_map_row
Relative Address	0x00000044
Absolute Address	0xF8006044
Width	28 bits
Access Type	rw
Reset Value	0xFF5555
Description	Select DRAM row address bits

Register DRAM_addr_map_row Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_addrmap_row_b15	27:24	rw	0xF	Selects the AXI address bit used as row address bit 15. Valid Range: 0 to 5, Internal Base: 24 The selected address bit is determined by adding the Internal Base to the value of this field. If set to 15, row address bit 15 is set to 0.
reg_ddrc_addrmap_row_b14	23:20	rw	0xF	Selects the AXI address bit used as row address bit 14. Valid Range: 0 to 6, Internal Base: 23 The selected address bit is determined by adding the Internal Base to the value of this field. If set to 15, row address bit 14 is set to 0.

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_addrmap_row_b13	19:16	rw	0x5	Selects the AXI address bit used as row address bit 13. Valid Range: 0 to 7, Internal Base: 22 The selected address bit is determined by adding the Internal Base to the value of this field. If set to 15, row address bit 13 is set to 0.
reg_ddrc_addrmap_row_b12	15:12	rw	0x5	Selects the AXI address bit used as row address bit 12. Valid Range: 0 to 8, Internal Base: 21 The selected address bit is determined by adding the Internal Base to the value of this field. If set to 15, row address bit 12 is set to 0.
reg_ddrc_addrmap_row_b2_11	11:8	rw	0x5	Selects the AXI address bits used as row address bits 2 to 11. Valid Range: 0 to 11. Internal Base: 11 (for row address bit 2) to 20 (for row address bit 11) The selected address bit for each of the row address bits is determined by adding the Internal Base to the value of this field.
reg_ddrc_addrmap_row_b1	7:4	rw	0x5	Selects the AXI address bits used as row address bit 1. Valid Range: 0 to 11. Internal Base: 10 The selected address bit for each of the row address bits is determined by adding the Internal Base to the value of this field.
reg_ddrc_addrmap_row_b0	3:0	rw	0x5	Selects the AXI address bits used as row address bit 0. Valid Range: 0 to 11. Internal Base: 9 The selected address bit for each of the row address bits is determined by adding the Internal Base to the value of this field

Note: address bits are relative to a byte address. For example, the value 0x0FFF6666 selects byte address bits [29:15] as row address bits in a 32-bit bus width configuration.

Register ([ddrc](#)) DRAM_ODT_reg

Name	DRAM_ODT_reg
Relative Address	0x00000048
Absolute Address	0xF8006048
Width	30 bits
Access Type	rw
Reset Value	0x00000249
Description	DRAM ODT control

Register DRAM_ODT_reg Details

Parts of this register are unused.

Field Name	Bits	Type	Reset Value	Description
reserved	29:27	rw	0x0	Reserved. Do not modify.
reserved	26:24	rw	0x0	Reserved. Do not modify.
reserved	23:21	rw	0x0	Reserved. Do not modify.
reserved	20:18	rw	0x0	Reserved. Do not modify.
reg_phy_idle_local_odt	17:16	rw	0x0	Value to drive on the 2-bit local_odt PHY outputs when output enable is not asserted and a read is not in progress. Typically this is the value required to disable termination to save power when idle.
reg_phy_wr_local_odt	15:14	rw	0x0	Value to drive on the 2-bit local_odt PHY outputs when write levelling is enabled for DQS.
reg_phy_rd_local_odt	13:12	rw	0x0	Value to drive on the 2-bit local_odt PHY outputs when output enable is not asserted and a read is in progress (where 'in progress' is defined as after a read command is issued and until all read data has been returned all the way to the controller.) Typically this is set to the value required to enable termination at the desired strength for read usage.
reserved	11:9	rw	0x1	Reserved. Do not modify.
reserved	8:6	rw	0x1	Reserved. Do not modify.
reserved	5:3	rw	0x1	Reserved. Do not modify.
reserved	2:0	rw	0x1	Reserved. Do not modify.

Register ([ddrc](#)) phy_dbg_reg

Name	phy_dbg_reg
Relative Address	0x0000004C
Absolute Address	0xF800604C
Width	20 bits
Access Type	ro
Reset Value	0x00000000
Description	PHY debug

Register phy_dbg_reg Details

Field Name	Bits	Type	Reset Value	Description
phy_reg_bc_fifo_re3	19	ro	0x0	Debug read capture FIFO read enable for data slice 3.
phy_reg_bc_fifo_we3	18	ro	0x0	Debug read capture FIFO write enable, for data slice 3.

Field Name	Bits	Type	Reset Value	Description
phy_reg_bc_dqs_oe3	17	ro	0x0	Debug DQS output enable for data slice 3.
phy_reg_bc_dq_oe3	16	ro	0x0	Debug DQ output enable for data slice 3.
phy_reg_bc_fifo_re2	15	ro	0x0	Debug read capture FIFO read enable for data slice 2.
phy_reg_bc_fifo_we2	14	ro	0x0	Debug read capture FIFO write enable, for data slice 2.
phy_reg_bc_dqs_oe2	13	ro	0x0	Debug DQS output enable for data slice 2.
phy_reg_bc_dq_oe2	12	ro	0x0	Debug DQ output enable for data slice 2.
phy_reg_bc_fifo_re1	11	ro	0x0	Debug read capture FIFO read enable for data slice 1.
phy_reg_bc_fifo_we1	10	ro	0x0	Debug read capture FIFO write enable, for data slice 1.
phy_reg_bc_dqs_oe1	9	ro	0x0	Debug DQS output enable for data slice 1.
phy_reg_bc_dq_oe1	8	ro	0x0	Debug DQ output enable for data slice 1.
phy_reg_bc_fifo_re0	7	ro	0x0	Debug read capture FIFO read enable for data slice 0.
phy_reg_bc_fifo_we0	6	ro	0x0	Debug read capture FIFO write enable, for data slice 0.
phy_reg_bc_dqs_oe0	5	ro	0x0	Debug DQS output enable for data slice 0.
phy_reg_bc_dq_oe0	4	ro	0x0	Debug DQ output enable for data slice 0.
phy_reg_rdc_fifo_rst_err_cnt	3:0	ro	0x0	Counter for counting how many times the pointers of read capture FIFO differ when they are reset by dll_calib.

Register ([ddrc](#)) phy_cmd_timeout_rddata_cpt

Name	phy_cmd_timeout_rddata_cpt
Relative Address	0x00000050
Absolute Address	0xF8006050
Width	32 bits
Access Type	mixed
Reset Value	0x00010200
Description	PHY command time out and read data capture FIFO

Register phy_cmd_timeout_rddata_cpt Details

Field Name	Bits	Type	Reset Value	Description
reg_phy_wrlvl_num_of_dq0	31:28	rw	0x0	This register value determines register determines the number of samples used for each ratio increment during Write Leveling. Num_of_iteration = reg_phy_wrlvl_num_of_dq0 + 1 The recommended value for this register is 8. Accuracy is better with higher value, but this will cause leveling to run longer.
reg_phy_gatlvl_num_of_dq0	27:24	rw	0x0	This register value determines register determines the number of samples used for each ratio increment during Gate Training. Num_of_iteration = reg_phy_gatlvl_num_of_dq0 + 1 The recommended value for this register is 8. Accuracy is better with higher value, but this will cause leveling to run longer.
reserved	23:20	ro	0x0	Reserved
reg_phy_clk_stall_level	19	rw	0x0	1: stall clock, for DLL aging control
reg_phy_dis_phy_ctrl_rstn	18	rw	0x0	Disable the reset from Phy Ctrl macro. 1: PHY Ctrl macro reset port is always HIGH 0: PHY Ctrl macro gets power on reset.
reg_phy_rdc_fifo_rst_err_cnt_clr	17	rw	0x0	Clear/reset for counter rdc_fifo_rst_err_cnt[3:0]. 0: no clear, 1: clear. Note: This is a synchronous dynamic signal that must have timing closed.
reg_phy_use_fixed_re	16	rw	0x1	When 1: PHY generates FIFO read enable after fixed number of clock cycles as defined by reg_phy_rdc_we_to_re_delay[3:0]. When 0: PHY uses the not_empty method to do the read enable generation. Note: This port must be set HIGH during training/leveling process i.e. when ddrdc_dfi_wrlvl_en/ ddrdc_dfi_rdlvl_en/ ddrdc_dfi_rdlvl_gate_en port is set HIGH.
reg_phy_rdc_fifo_rst_disable	15	rw	0x0	When 1, disable counting the number of times the Read Data Capture FIFO has been reset when the FIFO was not empty.
reserved	14:12	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
reg_phy_rdc_we_to_re_delay	11:8	rw	0x2	This register value + 1 give the number of clock cycles between writing into the Read Capture FIFO and the read operation. The setting of this register determines the read data timing and depends upon total delay in the system for read operation which include fly-by delays, trace delay, clkout_invert etc. This is used only if reg_phy_use_fixed_re=1.
reg_phy_wr_cmd_to_data	7:4	rw	0x0	Not used in DFI PHY.
reg_phy_rd_cmd_to_data	3:0	rw	0x0	Not used in DFI PHY.

Register ([ddrc](#)) mode_sts_reg

Name	mode_sts_reg
Relative Address	0x00000054
Absolute Address	0xF8006054
Width	21 bits
Access Type	ro
Reset Value	0x00000000
Description	Controller operation mode status

Register mode_sts_reg Details

Field Name	Bits	Type	Reset Value	Description
ddrc_reg_dbg_hpr_q_depth	20:16	ro	0x0	Indicates the number of entries currently in the High Priority Read (HPR) CAM.
ddrc_reg_dbg_lpr_q_depth	15:10	ro	0x0	Indicates the number of entries currently in the Low Priority Read (LPR) CAM.
ddrc_reg_dbg_wr_q_depth	9:4	ro	0x0	Indicates the number of entries currently in the Write CAM.
ddrc_reg_dbg_stall	3	ro	0x0	0: commands are being accepted. 1: no commands are accepted by the controller.
ddrc_reg_operating_mode	2:0	ro	0x0	Gives the status of the controller. 0: DDRC Init 1: Normal operation 2: Powerdown mode 3: Self-refresh mode 4 and above: deep power down mode (LPDDR2 only)

Register ([ddrc](#)) DLL_calib

Name	DLL_calib
Relative Address	0x00000058
Absolute Address	0xF8006058
Width	17 bits
Access Type	rw
Reset Value	0x00000101
Description	DLL calibration

Register DLL_calib Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_dis_dll_calib	16	rw	0x0	When 1, disable dll_calib generated by the controller. The core should issue the dll_calib signal using co_gs_dll_calib input. This input is changeable on the fly. When 0, controller will issue dll_calib periodically
reserved	15:8	rw	0x1	Reserved. Do not modify.
reserved	7:0	rw	0x1	Reserved. Do not modify.

Register ([ddrc](#)) ODT_delay_hold

Name	ODT_delay_hold
Relative Address	0x0000005C
Absolute Address	0xF800605C
Width	16 bits
Access Type	rw
Reset Value	0x00000023
Description	ODT delay and ODT hold

Register ODT_delay_hold Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_wr_odt_hold	15:12	rw	0x0	Cycles to hold ODT for a Write Command. When 0x0, ODT signal is ON for 1 cycle. When 0x1, it is ON for 2 cycles, etc. The values to program in different modes are : DRAM Burst of 4 -2, DRAM Burst of 8 -4
reg_ddrc_rd_odt_hold	11:8	rw	0x0	Unused

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_wr_odt_delay	7:4	rw	0x2	The delay, in clock cycles, from issuing a write command to setting ODT values associated with that command. ODT setting should remain constant for the entire time that DQS is driven by the controller. The suggested value for DDR2 is WL - 5 and for DDR3 is 0. WL is Write latency. DDR2 ODT has a 2-cycle on-time delay and a 2.5-cycle off-time delay. ODT is not applicable to LPDDR2.
reg_ddrc_rd_odt_delay	3:0	rw	0x3	UNUSED

Register ([ddrc](#)) ctrl_reg1

Name	ctrl_reg1
Relative Address	0x00000060
Absolute Address	0xF8006060
Width	13 bits
Access Type	mixed
Reset Value	0x0000003E
Description	Controller 1

Register ctrl_reg1 Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_selfref_en	12	rw	0x0	If 1, then the controller will put the DRAM into self refresh when the transaction store is empty. Dynamic Bit Field.
reserved	11	ro	0x0	Always keep this set to 0x0
reg_ddrc_dis_collision_page_opt	10	rw	0x0	When this is set to 0, auto-precharge will be disabled for the flushed command in a collision case. Collision cases are write followed by read to same address, read followed by write to same address, or write followed by write to same address with DIS_WC bit = 1 (where 'same address' comparisons exclude the two address bits representing critical word).
reg_ddrc_dis_wc	9	rw	0x0	Disable Write Combine: 0: enable 1: disable

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_refresh_update_level	8	rw	0x0	Toggle this signal to indicate that refresh register(s) have been updated. The value will be automatically updated when exiting soft reset. So it does not need to be toggled initially. Dynamic Bit Field.
reg_ddrc_auto_pre_en	7	rw	0x0	When set, most reads and writes will be issued with auto-precharge. (Exceptions can be made for collision cases.)
reg_ddrc_lpr_num_entries	6:1	rw	0x1F	Number of entries in the low priority transaction store is this value plus 1. In this design, by default all read ports are treated as low priority and hence the value of 0x1F. The hpr_num_entries is 32 minus this value. Bit [6] is ignored.
reg_ddrc_pageclose	0	rw	0x0	If true, bank will be closed and kept closed if no transactions are available for it. If false, bank will remain open until there is a need to close it (to open a different page, or for page timeout or refresh timeout.) This does not apply when auto-refresh is used.

Register ([ddrc](#)) ctrl_reg2

Name	ctrl_reg2
Relative Address	0x00000064
Absolute Address	0xF8006064
Width	18 bits
Access Type	mixed
Reset Value	0x00020000
Description	Controller 2

Register ctrl_reg2 Details

Field Name	Bits	Type	Reset Value	Description
reg_arb_go2critical_en	17	rw	0x1	0: Keep reg_ddrc_go2critical_wr and reg_ddrc_go2critical_rd signals going to DDRC at 0. 1: Set reg_ddrc_go2critical_wr and reg_ddrc_go2critical_rd signals going to DDRC based on Urgent input coming from AXI master.
reserved	16:13	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_go2critical_hysteresis	12:5	rw	0x0	Describes the number of cycles that co_gs_go2critical_rd or co_gs_go2critical_wr must be asserted before the corresponding queue moves to the 'critical' state in the DDRC. The arbiter controls the co_gs_go2critical_* signals; it is designed for use with this hysteresis field set to 0.
reserved	4:0	ro	0x0	Reserved

Register ([ddrc](#)) ctrl_reg3

Name	ctrl_reg3
Relative Address	0x00000068
Absolute Address	0xF8006068
Width	26 bits
Access Type	rw
Reset Value	0x00284027
Description	Controller 3

Register ctrl_reg3 Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_dfi_t_wlmrdr	25:16	rw	0x28	DDR2 and LPDDR2: not applicable. DDR3: First DQS/DQS# rising edge after write leveling mode is programmed. This is same as the tMLRD value from the DRAM spec.
reg_ddrc_rdlvl_rr	15:8	rw	0x40	DDR2 and LPDDR2: not applicable. DDR3: Read leveling read-to-read delay. Specifies the minimum number of clock cycles from the assertion of a read command to the next read command. Only applicable when connecting to PHYs operating in PHY RdLvl Evaluation mode.
reg_ddrc_wrlvl_ww	7:0	rw	0x27	DDR2: not applicable. LPDDR2 and DDR3: Write leveling write-to-write delay. Specifies the minimum number of clock cycles from the assertion of a ddrc_dfi_wrlvl_strobe signal to the next ddrc_dfi_wrlvl_strobe signal. Only applicable when connecting to PHYs operating in PHY RdLvl Evaluation mode. Recommended value is: (RL + reg_phy_rdc_we_to_re_delay + 50)

Register ([ddrc](#)) ctrl_reg4

Name	ctrl_reg4
Relative Address	0x0000006C
Absolute Address	0xF800606C
Width	16 bits
Access Type	rw
Reset Value	0x00001610
Description	Controller 4

Register ctrl_reg4 Details

Field Name	Bits	Type	Reset Value	Description
dfi_t_ctrlupd_interval_max_x1024	15:8	rw	0x16	This is the maximum amount of time between Controller initiated DFI update requests. This timer resets with each update request; when the timer expires, traffic is blocked for a few cycles. PHY can use this idle time to recalibrate the delay lines to the DLLs. The DLL calibration is also used to reset PHY FIFO pointers in case of data capture errors. Updates are required to maintain calibration over PVT, but frequent updates may impact performance. Units: 1024 clocks
dfi_t_ctrlupd_interval_min_x1024	7:0	rw	0x10	This is the minimum amount of time between Controller initiated DFI update requests (which will be executed whenever the controller is idle). Set this number higher to reduce the frequency of update requests, which can have a small impact on the latency of the first read request when the controller is idle. Units: 1024 clocks

Register ([ddrc](#)) ctrl_reg5

Name	ctrl_reg5
Relative Address	0x00000078
Absolute Address	0xF8006078
Width	32 bits
Access Type	mixed
Reset Value	0x00455111
Description	Controller register 5

Register ctrl_reg5 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	ro	0x0	Reserved
reg_ddrc_t_ckesr	25:20	rw	0x4	Minimum CKE low width for Self Refresh entry to exit Timing in memory clock cycles. Recommended settings: LPDDR2: tCKESR DDR2: tCKE DDR3: tCKE+1
reg_ddrc_t_cksrx	19:16	rw	0x5	This is the time before Self Refresh Exit that CK is maintained as a valid clock before issuing SRX. Specifies the clock stable time before SRX. Recommended settings: LPDDR2: 2 DDR2: 1 DDR3: tCKSRX
reg_ddrc_t_cksre	15:12	rw	0x5	This is the time after Self Refresh Entry that CK is maintained as a valid clock. Specifies the clock disable delay after SRE. Recommended settings: LPDDR2: 2 DDR2: 1 DDR3: tCKSRE
reg_ddrc_dfi_t_dram_clk_enable	11:8	rw	0x1	Specifies the number of DFI clock cycles from the de-assertion of the ddrc_dfi_dram_clk_disable signal on the DFI until the first valid rising edge of the clock to the DRAM memory devices at the PHY-DRAM boundary. If the DFI clock and the memory clock are not phase aligned, this timing parameter should be rounded up to the next integer value.
reg_ddrc_dfi_t_dram_clk_disable	7:4	rw	0x1	Specifies the number of DFI clock cycles from the assertion of the ddrc_dfi_dram_clk_disable signal on the DFI until the clock to the DRAM memory devices, at the PHY-DRAM boundary, maintains a low value. If the DFI clock and the memory clock are not phase aligned, this timing parameter should be rounded up to the next integer value.
reg_ddrc_dfi_t_ctrl_delay	3:0	rw	0x1	Specifies the number of DFI clock cycles after an assertion or deassertion of the DFI control signals that the control signals at the PHY-DRAM interface reflect the assertion or de-assertion. If the DFI clock and the memory clock are not phase-aligned, this timing parameter should be rounded up to the next integer value.

Register ([ddrc](#)) ctrl_reg6

Name	ctrl_reg6
Relative Address	0x0000007C
Absolute Address	0xF800607C
Width	32 bits
Access Type	mixed
Reset Value	0x00032222
Description	Controller register 6

Register ctrl_reg6 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:20	ro	0x0	Reserved
reg_ddrc_t_ckcsx	19:16	rw	0x3	This is the time before Clock Stop Exit that CK is maintained as a valid clock before issuing DPDX. Specifies the clock stable time before next command after Clock Stop Exit. Recommended setting for LPDDR2: tXP + 2.
reg_ddrc_t_ckdpdx	15:12	rw	0x2	This is the time before Deep Power Down Exit that CK is maintained as a valid clock before issuing DPDX. Specifies the clock stable time before DPDX. Recommended setting for LPDDR2: 2.
reg_ddrc_t_ckdpde	11:8	rw	0x2	This is the time after Deep Power Down Entry that CK is maintained as a valid clock. Specifies the clock disable delay after DPDE. Recommended setting for LPDDR2: 2.
reg_ddrc_t_ckpdx	7:4	rw	0x2	This is the time before Power Down Exit that CK is maintained as a valid clock before issuing PDX. Specifies the clock stable time before PDX. Recommended setting for LPDDR2: 2.
reg_ddrc_t_ckpde	3:0	rw	0x2	This is the time after Power Down Entry that CK is maintained as a valid clock. Specifies the clock disable delay after PDE. Recommended setting for LPDDR2: 2.

Register ([ddrc](#)) CHE_REFRESH_TIMER01

Name	CHE_REFRESH_TIMER01
Relative Address	0x000000A0
Absolute Address	0xF80060A0
Width	24 bits

Access Type rw
Reset Value 0x00008000
Description CHE_REFRESH_TIMER01

Register CHE_REFRESH_TIMER01 Details

Field Name	Bits	Type	Reset Value	Description
reserved	23:12	rw	0x8	Reserved. Do not modify.
reserved	11:0	rw	0x0	Reserved. Do not modify.

Register ([ddrc](#)) CHE_T_ZQ

Name CHE_T_ZQ
Relative Address 0x000000A4
Absolute Address 0xF80060A4
Width 32 bits
Access Type rw
Reset Value 0x10300802
Description ZQ parameters

Register CHE_T_ZQ Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_t_zq_short_nop	31:22	rw	0x40	DDR2: not applicable. LPDDR2 and DDR3: Number of cycles of NOP required after a ZQCS (ZQ calibration short) command is issued to DRAM. Units: Clock cycles.
reg_ddrc_t_zq_long_nop	21:12	rw	0x300	DDR2: not applicable. LPDDR2 and DDR3: Number of cycles of NOP required after a ZQCL (ZQ calibration long) command is issued to DRAM. Units: Clock cycles.
reg_ddrc_t_mod	11:2	rw	0x200	Mode register set command update delay (minimum d'128)
reg_ddrc_ddr3	1	rw	0x1	Indicates operating in DDR2/DDR3 mode. Default value is set for DDR3.
reg_ddrc_dis_auto_zq	0	rw	0x0	1=disable controller generation of ZQCS command. Co_gs_zq_calib_short can be used instead to control ZQ calibration commands. 0=internally generate ZQCS commands based on reg_ddrc_t_zq_short_interval_x1024 This is only present for implementations supporting DDR3 and LPDDR2 devices.

Register ([ddrc](#)) CHE_T_ZQ_Short_Interval_Reg

Name	CHE_T_ZQ_Short_Interval_Reg
Relative Address	0x000000A8
Absolute Address	0xF80060A8
Width	28 bits
Access Type	rw
Reset Value	0x0020003A
Description	Misc parameters

Register CHE_T_ZQ_Short_Interval_Reg Details

Field Name	Bits	Type	Reset Value	Description
dram_rstn_x1024	27:20	rw	0x2	Number of cycles to assert DRAM reset signal during init sequence. Units: 1024 Clock cycles. Applicable for DDR3 only.
t_zq_short_interval_x1024	19:0	rw	0x3A	DDR2: not used. LPDDR2 and DDR3: Average interval to wait between automatically issuing ZQCS (ZQ calibration short) commands to DDR3 devices. Meaningless if reg_ddrc_dis_auto_zq=1. Units: 1024 Clock cycles.

Register ([ddrc](#)) deep_pwrdown_reg

Name	deep_pwrdown_reg
Relative Address	0x000000AC
Absolute Address	0xF80060AC
Width	9 bits
Access Type	rw
Reset Value	0x00000000
Description	Deep powerdown (LPDDR2)

Register deep_pwrdown_reg Details

Field Name	Bits	Type	Reset Value	Description
deppowerdown_to_x1024	8:1	rw	0x0	DDR2 and DDR3: not sued. LPDDR2: Minimum deep power down time. DDR exits from deep power down mode immediately after reg_ddrc_deppowerdown_en is deasserted. Value from the spec is 500us. Units are in 1024 clock cycles. For performance only.
deppowerdown_en	0	rw	0x0	DDR2 and DDR3: not used. LPDDR2: 0: Brings Controller out of Deep Powerdown mode. 1: Puts DRAM into Deep Powerdown mode when the transaction store is empty. For performance only. Dynamic Bit Field.

Register ([ddrc](#)) reg_2c

Name	reg_2c
Relative Address	0x000000B0
Absolute Address	0xF80060B0
Width	29 bits
Access Type	mixed
Reset Value	0x00000000
Description	Training control

Register reg_2c Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_dfi_rd_data_eye_train	28	rw	0x0	DDR2: not applicable. LPDDR2 and DDR3: 0: 1: Read Data Eye training mode has been enabled as part of init sequence.
reg_ddrc_dfi_rd_dqs_gate_level	27	rw	0x0	0: Read DQS gate leveling is disabled. 1: Read DQS Gate Leveling mode has been enabled as part of init sequence; Valid only for DDR3 DFI designs
reg_ddrc_dfi_wr_level_en	26	rw	0x0	0: Write leveling disabled. 1: Write leveling mode has been enabled as part of init sequence; Valid only for DDR3 DFI designs

Field Name	Bits	Type	Reset Value	Description
ddrc_reg_trdlvl_max_error	25	ro	0x0	DDR2: not applicable. LPDDR2 and DDR3: When '1' indicates that the reg_ddrc_dfi_rdlvl_max_x1024 timer has timed out. This is a Clear-on-Write register. If read leveling or gate training timed out, an error is indicated by the DDRC and this bit gets set. The value is held at that value until it is cleared. Clearing is done by writing a '0' to this register.
ddrc_reg_twrldvl_max_error	24	ro	0x0	When '1' indicates that the reg_ddrc_dfi_wrlvl_max_x1024 timer has timed out. This is a Clear-on-Write register. If write leveling timed out, an error is indicated by the DDRC and this bit gets set. The value is held until it is cleared. Clearing is done by writing a '0' to this register. Only present in designs that support DDR3.
dfi_rdlvl_max_x1024	23:12	rw	0x0	Read leveling maximum time. Specifies the maximum number of clock cycles that the controller will wait for a response (phy_dfi_rdlvl_resp) to a read leveling enable signal (ddrc_dfi_rdlvl_en or ddrc_dfi_rdlvl_gate_en). Only applicable when connecting to PHY's operating in 'PHY RdLvl Evaluation' mode. Typical value 0xFFF Units 1024 clocks
dfi_wrlvl_max_x1024	11:0	rw	0x0	Write leveling maximum time. Specifies the maximum number of clock cycles that the controller will wait for a response (phy_dfi_wrlvl_resp) to a write leveling enable signal (ddrc_dfi_wrlvl_en). Only applicable when connecting to PHY's operating in 'PHY WrLvl Evaluation' mode. Typical value 0xFFF Units 1024 clocks

Register ([ddrc](#)) reg_2d

Name	reg_2d
Relative Address	0x000000B4
Absolute Address	0xF80060B4
Width	11 bits
Access Type	rw
Reset Value	0x00000200
Description	Misc Debug

Register reg_2d Details

Field Name	Bits	Type	Reset Value	Description
reserved	10	rw	0x0	Reserved. Do not modify.
reg_ddrc_skip_oed	9	rw	0x1	This register must be kept at 1'b1. 1'b0 is NOT supported. 1: Indicates the controller to skip OCD adjustment step during DDR2 initialization. OCD_Default and OCD_Exit are performed instead. 0: Not supported.
reserved	8:0	rw	0x0	Reserved. Do not modify.

Register ([ddrc](#)) dfi_timing

Name	dfi_timing
Relative Address	0x000000B8
Absolute Address	0xF80060B8
Width	25 bits
Access Type	rw
Reset Value	0x00200067
Description	DFI timing

Register dfi_timing Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_dfi_t_ctrlup_max	24:15	rw	0x40	Specifies the maximum number of clock cycles that the ddrc_dfi_ctrlupd_req signal can assert.
reg_ddrc_dfi_t_ctrlup_min	14:5	rw	0x3	Specifies the minimum number of clock cycles that the ddrc_dfi_ctrlupd_req signal must be asserted.
reg_ddrc_dfi_t_rddata_en	4:0	rw	0x7	Time from the assertion of a READ command on the DFI interface to the assertion of the phy_dfi_rddata_en signal. DDR2 and DDR3: RL - 1 LPDDR: RL Where RL is read latency of DRAM.

Register ([ddrc](#)) CHE_ECC_CONTROL_REG_OFFSET

Name	CHE_ECC_CONTROL_REG_OFFSET
Relative Address	0x000000C4
Absolute Address	0xF80060C4

Width	2 bits
Access Type	rw
Reset Value	0x00000000
Description	ECC error clear

Register CHE_ECC_CONTROL_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
Clear_Correctable_DRAM_ECC_error	1	rw	0x0	Writing 1 to this bit will clear the correctable log valid bit and the correctable error counters.
Clear_Uncorrectable_DRAM_ECC_error	0	rw	0x0	Writing 1 to this bit will clear the uncorrectable log valid bit and the uncorrectable error counters.

Register ([ddrc](#)) CHE_CORR_ECC_LOG_REG_OFFSET

Name	CHE_CORR_ECC_LOG_REG_OFFSET
Relative Address	0x000000C8
Absolute Address	0xF80060C8
Width	8 bits
Access Type	mixed
Reset Value	0x00000000
Description	ECC error correction

Register CHE_CORR_ECC_LOG_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
ECC_CORRECTED_BIT_NUM	7:1	clon wr	0x0	<p>Indicator of the bit number syndrome in error for single-bit errors. The field is 7-bit wide to handle 72-bits of data.</p> <p>This is an encoded value with ECC bits placed in between data. The encoding is given in section 5.4 Correctable bit number from the lowest error lane is reported here. There are only 13-valid bits going to an ECC lane (8-data + 5-ECC). Only 4-bits are needed to encode a max value of d'13. Bit[7] of this register is used to indicate the exact byte lane. When a error happens, if CORR_ECC_LOG_COL[0] from register 0x33 is 1'b0, then the error happened in Lane 0 or 1. If CORR_ECC_LOG_COL[0] is 1'b1, then the error happened in Lane 2 or 3. Bit[7] of this register indicates whether the error is from upper or lower byte lane. If it is 0, then it is lower byte lane and if it is 1, then it is upper byte lane. Together with CORR_ECC_LOG_COL[0] and bit[7] of this register, the exact byte lane with correctable error can be determined.</p>
CORR_ECC_LOG_VALID	0	ro	0x0	Set to 1 when a correctable ECC error is captured. As long as this is 1 no further ECC errors will be captured. This is cleared when a 1 is written to register bit[1] of ECC CONTROL REGISTER (0x31)

Register ([ddrc](#)) CHE_CORR_ECC_ADDR_REG_OFFSET

Name	CHE_CORR_ECC_ADDR_REG_OFFSET
Relative Address	0x000000CC
Absolute Address	0xF80060CC
Width	31 bits
Access Type	ro
Reset Value	0x00000000
Description	ECC error correction address log

Register CHE_CORR_ECC_ADDR_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
CORR_ECC_LOG_BANK	30:28	ro	0x0	Bank [2:0]

Field Name	Bits	Type	Reset Value	Description
CORR_ECC_LOG_ROW	27:12	ro	0x0	Row [15:0]
CORR_ECC_LOG_COLUMN	11:0	ro	0x0	Column [11:0]

Register ([ddrc](#)) CHE_CORR_ECC_DATA_31_0_REG_OFFSET

Name	CHE_CORR_ECC_DATA_31_0_REG_OFFSET
Relative Address	0x000000D0
Absolute Address	0xF80060D0
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	ECC error correction data log low

Register CHE_CORR_ECC_DATA_31_0_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
CORR_ECC_LOG_DATA_31_0	31:0	ro	0x0	Bits [31:0] of the data that caused the captured correctable ECC error. (Data associated with the first ECC error if multiple errors occurred since CORR_ECC_LOG_VALID was cleared). Since each ECC engine handles 8-bits of data, only the lower 8-bits of this register have valid data. The upper 24-bits will always be 0.

Register ([ddrc](#)) CHE_CORR_ECC_DATA_63_32_REG_OFFSET

Name	CHE_CORR_ECC_DATA_63_32_REG_OFFSET
Relative Address	0x000000D4
Absolute Address	0xF80060D4
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	ECC error correction data log mid

Register CHE_CORR_ECC_DATA_63_32_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
CORR_ECC_LOG_DATA_63_32	31:0	ro	0x0	Bits [63:32] of the data that caused the captured correctable ECC error. (Data associated with the first ECC error if multiple errors occurred since CORR_ECC_LOG_VALID was cleared) Since each ECC engine handles 8-bits of data and that is logged in register 0x34, all the 32-bits of this register will always be 0.

Register ([ddrc](#)) CHE_CORR_ECC_DATA_71_64_REG_OFFSET

Name	CHE_CORR_ECC_DATA_71_64_REG_OFFSET
Relative Address	0x000000D8
Absolute Address	0xF80060D8
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	ECC error correction data log high

Register CHE_CORR_ECC_DATA_71_64_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
CORR_ECC_LOG_DATA_71_64	7:0	ro	0x0	Bits [71:64] of the data that caused the captured correctable ECC error. (Data associated with the first ECC error if multiple errors occurred since CORR_ECC_LOG_VALID was cleared) 5-bits of ECC are calculated over 8-bits of data. Bits[68:64] carries these 5-bits. Bits[71:69] are always 0.

Register ([ddrc](#)) CHE_UNCORR_ECC_LOG_REG_OFFSET

Name	CHE_UNCORR_ECC_LOG_REG_OFFSET
Relative Address	0x000000DC
Absolute Address	0xF80060DC
Width	1 bits
Access Type	clonwr
Reset Value	0x00000000
Description	ECC unrecoverable error status

Register CHE_UNCORR_ECC_LOG_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
UNCORR_ECC_LOG_VALID	0	clon wr	0x0	Set to 1 when an uncorrectable ECC error is captured. As long as this is a 1, no further ECC errors will be captured. This is cleared when a 1 is written to register bit[0] of ECC CONTROL REGISTER (0x31).

Register ([ddrc](#)) CHE_UNCORR_ECC_ADDR_REG_OFFSET

Name	CHE_UNCORR_ECC_ADDR_REG_OFFSET
Relative Address	0x000000E0
Absolute Address	0xF80060E0
Width	31 bits
Access Type	ro
Reset Value	0x00000000
Description	ECC unrecoverable error address

Register CHE_UNCORR_ECC_ADDR_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
UNCORR_ECC_LOG_BANK	30:28	ro	0x0	Bank [2:0]
UNCORR_ECC_LOG_ROW	27:12	ro	0x0	Row [15:0]
UNCORR_ECC_LOG_COL	11:0	ro	0x0	Column [11:0]

Register ([ddrc](#)) CHE_UNCORR_ECC_DATA_31_0_REG_OFFSET

Name	CHE_UNCORR_ECC_DATA_31_0_REG_OFFSET
Relative Address	0x000000E4
Absolute Address	0xF80060E4
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	ECC unrecoverable error data low

Register CHE_UNCORR_ECC_DATA_31_0_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
UNCORR_ECC_LOG_DAT_31_0	31:0	ro	0x0	bits [31:0] of the data that caused the captured uncorrectable ECC error. (Data associated with the first ECC error if multiple errors occurred since UNCORR_ECC_LOG_VALID was cleared). Since each ECC engine handles 8-bits of data, only the lower 8-bits of this register have valid data. The upper 24-bits will always be 0.

Register ([ddrc](#)) CHE_UNCORR_ECC_DATA_63_32_REG_OFFSET

Name	CHE_UNCORR_ECC_DATA_63_32_REG_OFFSET
Relative Address	0x000000E8
Absolute Address	0xF80060E8
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	ECC unrecoverable error data middle

Register CHE_UNCORR_ECC_DATA_63_32_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
UNCORR_ECC_LOG_DAT_63_32	31:0	ro	0x0	bits [63:32] of the data that caused the captured uncorrectable ECC error. (Data associated with the first ECC error if multiple errors occurred since CORR_ECC_LOG_VALID was cleared) Since each ECC engine handles 8-bits of data and that is logged in register 0x34, all the 32-bits of this register will always be 0.

Register ([ddrc](#)) CHE_UNCORR_ECC_DATA_71_64_REG_OFFSET

Name	CHE_UNCORR_ECC_DATA_71_64_REG_OFFSET
Relative Address	0x000000EC
Absolute Address	0xF80060EC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	ECC unrecoverable error data high

Register CHE_UNCORR_ECC_DATA_71_64_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
UNCORR_ECC_LOG_DAT_71_64	7:0	ro	0x0	bits [71:64] of the data that caused the captured uncorrectable ECC error. (Data associated with the first ECC error if multiple errors occurred since UNCORR_ECC_LOG_VALID was cleared) 5-bits of ECC are calculated over 8-bits of data. Bits[68:64] carries these 5-bits. Bits[71:69] are always 0.

Register ([ddrc](#)) CHE_ECC_STATS_REG_OFFSET

Name	CHE_ECC_STATS_REG_OFFSET
Relative Address	0x000000F0
Absolute Address	0xF80060F0
Width	16 bits
Access Type	clonwr
Reset Value	0x00000000
Description	ECC error count

Register CHE_ECC_STATS_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
STAT_NUM_CORR_ERR	15:8	clonwr	0x0	Returns the number of correctable ECC errors seen since the last read. Counter saturates at max value. This is cleared when a 1 is written to register bit[1] of ECC CONTROL REGISTER (0x58).
STAT_NUM_UNCORR_ERR	7:0	clonwr	0x0	Returns the number of uncorrectable errors since the last read. Counter saturates at max value. This is cleared when a 1 is written to register bit[0] of ECC CONTROL REGISTER (0x58).

Register ([ddrc](#)) ECC_scrub

Name	ECC_scrub
Relative Address	0x000000F4
Absolute Address	0xF80060F4
Width	4 bits
Access Type	rw
Reset Value	0x00000008
Description	ECC mode/scrub

Register ECC_scrub Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_dis_scrub	3	rw	0x1	0: Enable ECC scrubs (valid only when reg_ddrc_ecc_mode = 100). 1: Disable ECC scrubs
reg_ddrc_ecc_mode	2:0	rw	0x0	DRAM ECC Mode. The only valid values that works for this project are 000 (No ECC) and 100 (SEC/DED over 1-beat). To run the design in ECC mode, set reg_ddrc_data_bus_width to 2'b01 (Half bus width) and reg_ddrc_ecc_mode to 100. In this mode, there will be 16-data bits + 6-bit ECC on the DRAM bus. Controller must NOT be put in full bus width mode, when ECC is turned ON. 000 : No ECC, 001: Reserved 010: Parity 011: Reserved 100: SEC/DED over 1-beat 101: SEC/DED over multiple beats 110: Device Correction 111: Reserved

Register ([ddrc](#)) CHE_ECC_CORR_BIT_MASK_31_0_REG_OFFSET

Name	CHE_ECC_CORR_BIT_MASK_31_0_REG_OFFSET
Relative Address	0x000000F8
Absolute Address	0xF80060F8
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	ECC data mask low

Register CHE_ECC_CORR_BIT_MASK_31_0_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
ddrc_reg_ecc_corr_bit_mask	31:0	ro	0x0	Bits [31:0] of ddrc_reg_ecc_corr_bit_mask register. Indicates the mask of the corrected data. 1 - on any bit indicates that the bit has been corrected by the DRAM ECC logic 0 - on any bit indicates that the bit has NOT been corrected by the DRAM ECC logic. Valid when any bit of 'ddrc_reg_ecc_corrected_err' is high. This mask doesn't indicate any correction that has been made in the ECC check bits. If there are errors in multiple lanes, then this signal will have the mask for the lowest lane. Each ECC engine works on 8-bits of data. Hence only the lower 8-bits carry valid information. Upper 24-bits are always 0.

Register ([ddrc](#)) CHE_ECC_CORR_BIT_MASK_63_32_REG_OFFSET

Name	CHE_ECC_CORR_BIT_MASK_63_32_REG_OFFSET
Relative Address	0x000000FC
Absolute Address	0xF80060FC
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	ECC data mask high

Register CHE_ECC_CORR_BIT_MASK_63_32_REG_OFFSET Details

Field Name	Bits	Type	Reset Value	Description
ddrc_reg_ecc_corr_bit_mask	31:0	ro	0x0	Bits [63:32] of ddrc_reg_ecc_corr_bit_mask register. Indicates the mask of the corrected data. 1 - on any bit indicates that the bit has been corrected by the DRAM ECC logic 0 - on any bit indicates that the bit has NOT been corrected by the DRAM ECC logic. Valid when any bit of 'ddrc_reg_ecc_corrected_err' is high. This mask doesn't indicate any correction that has been made in the ECC check bits. If there are errors in multiple lanes, then this signal will have the mask for the lowest lane. Each ECC engine works on 8-bits of data and this is reported in register 0x3E. All 32-bits of this register are 0 always.

Register ([ddrc](#)) phy_rcvr_enable

Name	phy_rcvr_enable
------	-----------------

Relative Address	0x00000114
Absolute Address	0xF8006114
Width	8 bits
Access Type	rw
Reset Value	0x00000000
Description	Phy receiver enable register

Register phy_rcvr_enable Details

Field Name	Bits	Type	Reset Value	Description
reg_phy_dif_off	7:4	rw	0x0	Value to drive to IO receiver enable pins when turning it OFF. When in powerdown or self-refresh (CKE=0) this value will be sent to the IOs to control receiver on/off. IOD is the size specified by the IO_DIFEN_SIZE parameter. Depending on the IO, one of these signals dif_on or dif_off can be used.
reg_phy_dif_on	3:0	rw	0x0	Value to drive to IO receiver enable pins when turning it ON. When NOT in powerdown or self-refresh (when CKE=1) this value will be sent to the IOs to control receiver on/off. IOD is the size specified by the IO_DIFEN_SIZE parameter.

Register ([ddrc](#)) PHY_Config0

Name	PHY_Config0
Relative Address	0x00000118
Absolute Address	0xF8006118
Width	31 bits
Access Type	rw
Reset Value	0x40000001
Description	PHY configuration register for data slice 0.

Note: This register is the first in an array of 4 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
PHY_Config0	0xf8006118
PHY_Config1	0xf800611c
PHY_Config2	0xf8006120
PHY_Config3	0xf8006124

Register PHY_Config0 to PHY_Config3 Details

Field Name	Bits	Type	Reset Value	Description
reg_phy_dq_offset	30:24	rw	0x40	Offset value from DQS to DQ. Default value: 0x40 (for 90 degree shift). This is only used when reg_phy_use_wr_level=1. #Note 1: When a port width (W) is multiple of N instances of Ranks or Slices, each instance will get W/N bits. Instance n will get (n+1)*(W/N) -1: n (W/N) bits where n (0, 1, to N-1) is the instance number of Rank or Slice.
reg_phy_bist_err_clr	23:15	rw	0x0	Clear the mismatch error flag from the BIST Checker. 0: No effect 1: sticky error flag is cleared
reg_phy_bist_shift_dq	14:6	rw	0x0	Determines whether early shifting is required for a particular DQ bit when reg_phy_bist_mode is 10. 0: PRBS pattern without any shift. 1: PRBS pattern shifted early by 1 bit.
reserved	5	rw	0x0	Reserved. Do not modify.
reserved	4	rw	0x0	Reserved. Do not modify.
reg_phy_wrlvl_inc_mode	3	rw	0x0	reserved
reg_phy_gatlvl_inc_mode	2	rw	0x0	reserved
reg_phy_rdlvl_inc_mode	1	rw	0x0	reserved
reg_phy_data_slice_in_use	0	rw	0x1	Data bus width selection for Read FIFO RE generation. One bit for each data slice. 0: read data responses are ignored. 1: data slice is valid. Note: The Phy Data Slice 0 must always be enabled.

Register ([ddrc](#)) phy_init_ratio0

Name	phy_init_ratio0
Relative Address	0x0000012C
Absolute Address	0xF800612C
Width	20 bits
Access Type	rw
Reset Value	0x00000000
Description	PHY init ratio register for data slice 0.

Note: This register is the first in an array of 4 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
phy_init_ratio0	0xf800612c
phy_init_ratio1	0xf8006130
phy_init_ratio2	0xf8006134
phy_init_ratio3	0xf8006138

Register phy_init_ratio0 to phy_init_ratio3 Details

Field Name	Bits	Type	Reset Value	Description
reg_phy_gatlvl_init_ratio	19:10	rw	0x0	The user programmable init ratio used Gate Leveling FSM
reg_phy_wrlvl_init_ratio	9:0	rw	0x0	The user programmable init ratio used by Write Leveling FSM

Register ([ddrc](#)) phy_rd_dqs_cfg0

Name	phy_rd_dqs_cfg0
Relative Address	0x00000140
Absolute Address	0xF8006140
Width	20 bits
Access Type	rw
Reset Value	0x00000040
Description	PHY read DQS configuration register for data slice 0.

Note: This register is the first in an array of 4 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
phy_rd_dqs_cfg0	0xf8006140
phy_rd_dqs_cfg1	0xf8006144
phy_rd_dqs_cfg2	0xf8006148
phy_rd_dqs_cfg3	0xf800614c

Register phy_rd_dqs_cfg0 to phy_rd_dqs_cfg3 Details

Field Name	Bits	Type	Reset Value	Description
reg_phy_rd_dqs_slave_delay	19:11	rw	0x0	If reg_phy_rd_dqs_slave_force is 1, replace delay/tap value for read DQS slave DLL with this value.
reg_phy_rd_dqs_slave_force	10	rw	0x0	0: 1: overwrite the delay/tap value for read DQS slave DLL with the value of the debug_rd_dqs_slave_delay bus.
reg_phy_rd_dqs_slave_ratio	9:0	rw	0x40	Ratio value for read DQS slave DLL. This is the fraction of a clock cycle represented by the shift to be applied to the read DQS in units of 256ths. In other words, the full-cycle tap value from the master DLL will be scaled by this number over 256 to get the delay value for the slave delay line. Provide a default value of 0x40 for most applications

Register ([ddrc](#)) phy_wr_dqs_cfg0

Name	phy_wr_dqs_cfg0
Relative Address	0x00000154
Absolute Address	0xF8006154
Width	20 bits
Access Type	rw
Reset Value	0x00000000
Description	PHY write DQS configuration register for data slice 0.

Note: This register is the first in an array of 4 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
phy_wr_dqs_cfg0	0xf8006154
phy_wr_dqs_cfg1	0xf8006158

Name	Address
phy_wr_dqs_cfg2	0xf800615c
phy_wr_dqs_cfg3	0xf8006160

Register phy_wr_dqs_cfg0 to phy_wr_dqs_cfg3 Details

Field Name	Bits	Type	Reset Value	Description
reg_phy_wr_dqs_slave_delay	19:11	rw	0x0	If reg_phy_wr_dqs_slave_force is 1, replace delay/tap value for write DQS slave DLL with this value.
reg_phy_wr_dqs_slave_force	10	rw	0x0	0: 1: overwrite the delay/tap value for write DQS slave DLL with the value of the reg_phy_wr_dqs_slave_delay bus.
reg_phy_wr_dqs_slave_ratio	9:0	rw	0x0	Ratio value for write DQS slave DLL. This is the fraction of a clock cycle represented by the shift to be applied to the write DQS in units of 256ths. In other words, the full-cycle tap value from the master DLL will be scaled by this number over 256 to get the delay value for the slave delay line.

Register ([ddrc](#)) phy_we_cfg0

Name	phy_we_cfg0
Relative Address	0x00000168
Absolute Address	0xF8006168
Width	21 bits
Access Type	rw
Reset Value	0x00000040
Description	PHY FIFO write enable configuration for data slice 0.

Note: This register is the first in an array of 4 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
phy_we_cfg0	0xf8006168
phy_we_cfg1	0xf800616c
phy_we_cfg2	0xf8006170
phy_we_cfg3	0xf8006174

Register phy_we_cfg0 to phy_we_cfg3 Details

Field Name	Bits	Type	Reset Value	Description
reg_phy_fifo_we_in_delay	20:12	rw	0x0	Delay value to be used when debug_fifo_we_in_forceX is set to 1. R is the number of Ranks supported.
reserved	11	rw	0x0	Reserved. Do not modify.
reserved	10:0	rw	0x40	Reserved. Do not modify.

Register ([ddrc](#)) wr_data_slv0

Name	wr_data_slv0
Relative Address	0x0000017C
Absolute Address	0xF800617C
Width	20 bits
Access Type	rw
Reset Value	0x00000080
Description	PHY write data slave ratio config for data slice 0.

Note: This register is the first in an array of 4 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
wr_data_slv0	0xf800617c
wr_data_slv1	0xf8006180
wr_data_slv2	0xf8006184
wr_data_slv3	0xf8006188

Register wr_data_slv0 to wr_data_slv3 Details

Field Name	Bits	Type	Reset Value	Description
reg_phy_wr_data_slave_delay	19:11	rw	0x0	If reg_phy_wr_data_slave_force is 1, replace delay/tap value for write data slave DLL with this value.

Field Name	Bits	Type	Reset Value	Description
reg_phy_wr_data_slave_force	10	rw	0x0	0: 1: overwrite the delay / tap value for write data slave DLL with the value of the reg_phy_wr_data_slave_force bus.
reg_phy_wr_data_slave_ratio	9:0	rw	0x80	Ratio value for write data slave DLL. This is the fraction of a clock cycle represented by the shift to be applied to the write DQ muxes in units of 256ths. In other words, the full-cycle tap value from the master DLL will be scaled by this number over 256 to get the delay value for the slave delay line.

Register ([ddrc](#)) reg_64

Name	reg_64
Relative Address	0x00000190
Absolute Address	0xF8006190
Width	32 bits
Access Type	rw
Reset Value	0x10020000
Description	Training control 2

Register reg_64 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	rw	0x0	Reserved. Do not modify.
reg_phy_cmd_latency	30	rw	0x0	If set to 1, command comes to phy_ctrl through a flop.
reg_phy_lpddr	29	rw	0x0	0: DDR2 or DDR3. 1: LPDDR2.
reserved	28	rw	0x1	Reserved. Do not modify.
reg_phy_ctrl_slave_delay	27:21	rw	0x0	If reg_phy_rd_dqs_slave_force is 1, replace delay / tap value for address / command timing slave DLL with this value. This is a bit value, the remaining 2 bits are in register 0x65 bits[19:18].
reg_phy_ctrl_slave_force	20	rw	0x0	1: overwrite the delay / tap value for address / command timing slave DLL with the value of the reg_phy_rd_dqs_slave_delay bus.

Field Name	Bits	Type	Reset Value	Description
reg_phy_ctrl_slave_ratio	19:10	rw	0x80	Ratio value for address/command launch timing in phy_ctrl macro. This is the fraction of a clock cycle represented by the shift to be applied to the read DQS in units of 256ths. In other words, the full cycle tap value from the master DLL will be scaled by this number over 256 to get the delay value for the slave delay line.
reg_phy_sel_logic	9	rw	0x0	Selects one of the two read leveling algorithms. 'b0': Select algorithm # 1 'b1': Select algorithm # 2 Please refer to Read Data Eye Training section in PHY User Guide for details about the Read Leveling algorithms
reserved	8	rw	0x0	Reserved. Do not modify.
reg_phy_invert_clkout	7	rw	0x0	Inverts the polarity of DRAM clock. 0: core clock is passed on to DRAM 1: inverted core clock is passed on to DRAM. Use this when CLK can arrive at a DRAM device ahead of DQS or coincidence with DQS based on board topology. This effectively delays the CLK to the DRAM device by half -cycle, providing a CLK edge that DQS can align to during leveling.
reg_phy_bist_mode	6:5	rw	0x0	The mode bits select the pattern type generated by the BIST generator. All the patterns are transmitted continuously once enabled. 00: constant pattern (0 repeated on each DQ bit) 01: low freq pattern (00001111 repeated on each DQ bit) 10: PRBS pattern (2 ⁷ -1 PRBS pattern repeated on each DQ bit) Each DQ bit always has same data value except when early shifting in PRBS mode is requested 11: reserved
reg_phy_bist_force_err	4	rw	0x0	This register bit is used to check that BIST checker is not giving false pass. When this port is set 1, data bit gets inverted before sending out to the external memory and BIST checker must return a mismatch error.
reg_phy_bist_enable	3	rw	0x0	Enable the internal BIST generation and checker logic when this port is set HIGH. Setting this port as 0 will stop the BIST generator/checker. In order to run BIST tests, this port must be set along with reg_phy_loopback.

Field Name	Bits	Type	Reset Value	Description
reg_phy_at_spd_atpg	2	rw	0x0	0: run scan test at slow clock speed but with high coverage 1: run scan test at full clock speed but with less coverage During normal function mode, this port must be set 0.
reg_phy_bl2	1	rw	0x0	Reserved for future Use.
reserved	0	rw	0x0	Reserved. Do not modify.

Register ([ddrc](#)) reg_65

Name	reg_65
Relative Address	0x00000194
Absolute Address	0xF8006194
Width	20 bits
Access Type	rw
Reset Value	0x00000000
Description	Training control 3

Register reg_65 Details

Field Name	Bits	Type	Reset Value	Description
reg_phy_ctrl_slave_delay	19:18	rw	0x0	If reg-phy_rd_dqs_slave_force is 1, replace delay/tap value for address/command timing slave DLL with this value
reg_phy_dis_calib_rst	17	rw	0x0	Disable the dll_calib (internally generated) signal from resetting the Read Capture FIFO pointers and portions of phy_data. Note: dll_calib is (i) generated by dfi_ctrl_upd_req or (ii) by the PHY when it detects that the clock frequency variation has exceeded the bounds set by reg_phy_dll_lock_diff or (iii) periodically throughout the leveling process. dll_calib will update the slave DL with PVT-compensated values according to master DLL outputs

Field Name	Bits	Type	Reset Value	Description
reg_phy_use_rd_data_eye_level	16	rw	0x0	Read Data Eye training control. 0: Use register programmed ratio values 1: Use ratio for delay line calculated by data eye leveling Note: This is a Synchronous dynamic signal that requires timing closure
reg_phy_use_rd_dqs_gate_level	15	rw	0x0	Read DQS Gate training control. 0: Use register programmed ratio values 1: Use ratio for delay line calculated by DQS gate leveling Note: This is a Synchronous dynamic signal that requires timing closure.
reg_phy_use_wr_level	14	rw	0x0	Write Leveling training control. 0: Use register programmed ratio values 1: Use ratio for delay line calculated by write leveling Note: This is a Synchronous dynamic signal that requires timing closure.
reg_phy_dll_lock_diff	13:10	rw	0x0	The Maximum number of delay line taps variation allowed while maintaining the master DLL lock. When the PHY is in locked state and the variation on the clock exceeds the variation indicated by the register, the lock signal is deasserted
reg_phy_rd_rl_delay	9:5	rw	0x0	This delay determines when to select the active rank's ratio logic delay for Read Data and Read DQS slave delay lines after PHY receives a read command at Control Interface. The programmed value must be (Read Latency - 3) with a minimum value of 1.
reg_phy_wr_rl_delay	4:0	rw	0x0	This delay determines when to select the active rank's ratio logic delay for Write Data and Write DQS slave delay lines after PHY receives a write command at Control Interface. The programmed value must be (Write Latency - 4) with a minimum value of 1.

Register ([ddrc](#)) reg69_6a0

Name	reg69_6a0
Relative Address	0x000001A4
Absolute Address	0xF80061A4
Width	29 bits

Access Type	ro
Reset Value	0x000F0000
Description	Training results for data slice 0.

Note: This register is the first in an array of 2 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
reg69_6a0	0xf80061a4
reg69_6a1	0xf80061a8

Register reg69_6a0 to reg69_6a1 Details

Field Name	Bits	Type	Reset Value	Description
phy_reg_status_fifo_we_slave_dll_value	28:20	ro	0x0	Delay value applied to FIFO WE slave DLL.
phy_reg_rdlvl_fifo_we_ratio	19:9	ro	0x780	Ratio value generated by Read Gate training FSM.
phy_reg_bist_err	8:0	ro	0x0	Mismatch error flag from the BIST Checker. 1 bit per data slice. 1'b1: Pattern mismatch error 1'b0: All patterns matched This is a sticky flag. In order to clear this bit, port reg_phy_bist_err_clr must be set HIGH. Note that reg6b is unused.

Register ([ddrc](#)) reg6c_6d2

Name	reg6c_6d2
Relative Address	0x000001B0
Absolute Address	0xF80061B0
Width	29 bits
Access Type	ro
Reset Value	0x000F0000
Description	Training results for data slice 2.

Note: This register is the first in an array of 2 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
reg6c_6d2	0xf80061b0
reg6c_6d3	0xf80061b4

Register reg6c_6d2 to reg6c_6d3 Details

Field Name	Bits	Type	Reset Value	Description
phy_reg_status_fifo_we_slave_dll_value	28:20	ro	0x0	Delay value applied to FIFO WE slave DLL.
phy_reg_rdlvl_fifowein_ratio	19:9	ro	0x780	Ratio value generated by Read Gate training FSM.
phy_reg_bist_err	8:0	ro	0x0	Mismatch error flag from the BIST Checker. 1 bit per data slice. 1'b1: Pattern mismatch error 1'b0: All patterns matched This is a sticky flag. In order to clear this bit, port reg_phy_bist_err_clr must be set HIGH. Note that reg6b is unused.

Register ([ddrc](#)) reg6e_710

Name	reg6e_710
Relative Address	0x000001B8
Absolute Address	0xF80061B8
Width	30 bits
Access Type	ro
Reset Value	x
Description	Training results (2) for data slice 0.

Note: This register is the first in an array of 4 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
reg6e_710	0xf80061b8
reg6e_711	0xf80061bc
reg6e_712	0xf80061c0
reg6e_713	0xf80061c4

Register reg6e_710 to reg6e_713 Details

Field Name	Bits	Type	Reset Value	Description
phy_reg_rdlvl_dqs_ratio	29:20	ro	x	Ratio value generated by Read Data Eye training FSM.
phy_reg_wrlvl_dq_ratio	19:10	ro	x	Ratio value generated by the write leveling FSM for Write Data.
phy_reg_wrlvl_dqs_ratio	9:0	ro	x	Ratio value generated by the write leveling FSM for Write DQS.

Register ([ddrc](#)) phy_dll_sts0

Name	phy_dll_sts0
Relative Address	0x000001CC
Absolute Address	0xF80061CC
Width	27 bits
Access Type	ro
Reset Value	0x00000000
Description	Slave DLL results for data slice 0.

Note: This register is the first in an array of 4 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
phy_dll_sts0	0xf80061cc
phy_dll_sts1	0xf80061d0
phy_dll_sts2	0xf80061d4
phy_dll_sts3	0xf80061d8

Register phy_dll_sts0 to phy_dll_sts3 Details

Field Name	Bits	Type	Reset Value	Description
phy_reg_status_wr_dqs_slave_dll_value	26:18	ro	0x0	Delay value applied to write DQS slave DLL
phy_reg_status_wr_data_slave_dll_value	17:9	ro	0x0	Delay value applied to write data slave DLL
phy_reg_status_rd_dqs_slave_dll_value	8:0	ro	0x0	Delay value applied to read data slave DLL

Register ([ddrc](#)) dll_lock_sts

Name	dll_lock_sts
Relative Address	0x000001E0
Absolute Address	0xF80061E0
Width	24 bits
Access Type	ro
Reset Value	0x00000000
Description	DLL Lock Status, read

Register dll_lock_sts Details

Field Name	Bits	Type	Reset Value	Description
phy_reg_status_of_in_lock_state_1	23:22	ro	0x0	Lock status from the Output Filter module inside the Master DLL. Bit[0] - Fine delay line lock status. 1: locked, 0: unlocked. Bit[1] - Coarse delay line lock status. 1: locked, 0: unlocked. (For Master DLL 1)
phy_reg_status_of_in_lock_state_0	21:20	ro	0x0	Lock status from the Output Filter module inside the Master DLL. Bit[0] - Fine delay line lock status. 1: locked, 0: unlocked. Bit[1] - Coarse delay line lock status. 1: locked, 0: unlocked. (For Master DLL 0)
phy_reg_status_dll_slave_value_1	19:11	ro	0x0	Shows the current Coarse and Fine delay values going to all the Slave DLLs [1:0] - Fine value [8:2] - Coarse value (For Master DLL 1)
phy_reg_status_dll_slave_value_0	10:2	ro	0x0	Shows the current Coarse and Fine delay values going to all the Slave DLLs [1:0] - Fine value [8:2] - Coarse value (For Master DLL 0)
phy_reg_status_dll_lock_1	1	ro	0x0	Status Master DLL 1 signal: 0: not locked 1: locked
phy_reg_status_dll_lock_0	0	ro	0x0	Master DLL 0 Status signal: 0: not locked 1: locked

Register ([ddrc](#)) phy_ctrl_sts

Name	phy_ctrl_sts
Relative Address	0x000001E4
Absolute Address	0xF80061E4
Width	30 bits
Access Type	ro
Reset Value	x
Description	PHY Control status, read

Register phy_ctrl_sts Details

Field Name	Bits	Type	Reset Value	Description
phy_reg_status_phy_ctrl_of_in_lock_state	29:28	ro	0x0	Lock status from Master DLL Output Filter. 0: not locked, 1: locked. Bit 28: Fine delay line. Bit 29: Coarse delay line.
phy_reg_status_phy_ctrl_dll_slave_value	27:20	ro	0x0	Values applied to the PHY_CTRL Slave DLL: Bit field 21:20 is the Fine value Bit field 27:22 is the Course value
phy_reg_status_phy_ctrl_dll_lock	19	ro	0x0	PHY Control Master DLL Status: 0: not locked, 1: locked
phy_reg_status_of_out_delay_value	18:10	ro	x	Values from Master DDL Output Filter (no default value). Bit field 11:10 is the Fine value Bit field 18:12 is the Coarse value
phy_reg_status_of_in_delay_value	9:0	ro	x	Values applied to Master DDL Output Filter (no default value): Bit field 1:0 is the Fine value Bit field 9:2 is the Coarse value

Register ([ddrc](#)) phy_ctrl_sts_reg2

Name	phy_ctrl_sts_reg2
Relative Address	0x000001E8
Absolute Address	0xF80061E8
Width	27 bits
Access Type	ro
Reset Value	0x00000000
Description	PHY Control status (2), read

Register phy_ctrl_sts_reg2 Details

Field Name	Bits	Type	Reset Value	Description
phy_reg_status_phy_ctrl_slave_dll_value	26:18	ro	0x0	Delay value applied to read DQS slave DLL.
reserved	17:9	ro	0x0	reserved
phy_reg_status_phy_ctrl_of_in_delay_value	8:0	ro	0x0	Values applied to Master DLL Output Filter: Bit field 1:0 is the Fine value Bit field 8:2 is the Coarse value

Register ([ddrc](#)) axi_id

Name	axi_id
Relative Address	0x00000200
Absolute Address	0xF8006200
Width	26 bits
Access Type	ro
Reset Value	0x00153042
Description	ID and revision information

Register axi_id Details

Field Name	Bits	Type	Reset Value	Description
reg_arb_rev_num	25:20	ro	0x1	Revision Number
reg_arb_prov_num	19:12	ro	0x53	Prov number
reg_arb_part_num	11:0	ro	0x42	Part Number

Register ([ddrc](#)) page_mask

Name	page_mask
Relative Address	0x00000204
Absolute Address	0xF8006204
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Page mask

Register page_mask Details

Field Name	Bits	Type	Reset Value	Description
reg_arb_page_addr_mask	31:0	rw	0x0	<p>Set this register based on the value programmed on the reg_ddrc_addrmap_* registers.</p> <p>Set the Column address bits to 0. Set the Page and Bank address bits to 1.</p> <p>This is used for calculating page_match inside the slave modules in Arbiter. The page_match is considered during the arbitration process. This mask applies to 64-bit address and not byte address.</p> <p>Setting this value to 0 disables transaction prioritization based on page/bank match.</p>

Register ([ddrc](#)) axi_priority_wr_port0

Name	axi_priority_wr_port0
Relative Address	0x00000208
Absolute Address	0xF8006208
Width	20 bits
Access Type	mixed
Reset Value	0x000803FF
Description	AXI Priority control for write port 0.

Note: This register is the first in an array of 4 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
axi_priority_wr_port0	0xf8006208
axi_priority_wr_port1	0xf800620c
axi_priority_wr_port2	0xf8006210
axi_priority_wr_port3	0xf8006214

Register axi_priority_wr_port0 to axi_priority_wr_port3 Details

Field Name	Bits	Type	Reset Value	Description
reserved	19	rw	0x1	Reserved. Do not modify.
reg_arb_dis_page_mat ch_wr_portn	18	rw	0x0	Disable the page match feature.
reg_arb_disable_urgent t_wr_portn	17	rw	0x0	Disable urgent for this Write Port.
reg_arb_disable_aging _wr_portn	16	rw	0x0	Disable aging for this Write Port.
reserved	15:10	ro	0x0	Reserved
reg_arb_pri_wr_portn	9:0	rw	0x3FF	Priority of this Write Port n. Value in this register used to load the aging counters (when respective port request is asserted and grant is generated to that port). These register can be reprogrammed to set priority of each port. Lower the value more will be priority given to the port. For example if 0x82 (port 0) value is set to 'h3FF, and 0x83 (port 1) is set to 'h0FF, and both port0 and port1 have requests, in this case port1 will get high priority and grant will be given to port1.

Register ([ddrc](#)) axi_priority_rd_port0

Name	axi_priority_rd_port0
Relative Address	0x00000218
Absolute Address	0xF8006218
Width	20 bits
Access Type	mixed
Reset Value	0x000003FF
Description	AXI Priority control for read port 0.

Note: This register is the first in an array of 4 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
axi_priority_rd_port0	0xf8006218
axi_priority_rd_port1	0xf800621c
axi_priority_rd_port2	0xf8006220
axi_priority_rd_port3	0xf8006224

Register axi_priority_rd_port0 to axi_priority_rd_port3 Details

Field Name	Bits	Type	Reset Value	Description
reg_arb_set_hpr_rd_portn	19	rw	0x0	Enable reads to be generated as HPR for this Read Port.
reg_arb_dis_page_match_rd_portn	18	rw	0x0	Disable the page match feature.
reg_arb_disable_urgent_rd_portn	17	rw	0x0	Disable urgent for this Read Port.
reg_arb_disable_aging_rd_portn	16	rw	0x0	Disable aging for this Read Port.
reserved	15:10	ro	0x0	Reserved. Do not modify.
reg_arb_pri_rd_portn	9:0	rw	0x3FF	Priority of this Read Port n. Value in this register used to load the aging counters (when respective port request is asserted and grant is generated to that port). These register can be reprogrammed to set priority of each port. Lower the value more will be priority given to the port. For example if 0x82 (port 0) value is set to 'h3FF, and 0x83 (port 1) is set to 'h0FF, and both port0 and port1 have requests, in this case port1 will get high priority and grant will be given to port1.

Register ([ddrc](#)) trusted_mem_cfg

Name	trusted_mem_cfg
Relative Address	0x00000290
Absolute Address	0xF8006290
Width	16 bits
Access Type	rw
Reset Value	0x00000000
Description	Trusted Memory configuration

Register trusted_mem_cfg Details

Field Name	Bits	Type	Reset Value	Description
reg_decpot	15:0	rw	0x0	Sets the memory regions to be secure or non-secure. The memory is divided into 64Mb sections. For a 1 Gb part, the number of regions is 16. 0 - non-secure region 1 - secure region Reserved. Not used any more.

Register ([ddrc](#)) excl_access_cfg0

Name	excl_access_cfg0
Relative Address	0x00000294
Absolute Address	0xF8006294
Width	18 bits
Access Type	rw
Reset Value	0x00000000
Description	Exclusive access configuration for port 0.

Note: This register is the first in an array of 4 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
excl_access_cfg0	0xf8006294
excl_access_cfg1	0xf8006298
excl_access_cfg2	0xf800629c
excl_access_cfg3	0xf80062a0

Register excl_access_cfg0 to excl_access_cfg3 Details

Field Name	Bits	Type	Reset Value	Description
reg_excl_acc_id1_port	17:9	rw	0x0	Reserved
reg_excl_acc_id0_port	8:0	rw	0x0	Reserved

Register ([ddrc](#)) mode_reg_read

Name	mode_reg_read
Relative Address	0x000002A4
Absolute Address	0xF80062A4
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Mode register read data

Register mode_reg_read Details

This registers is applicable only when LPDDR2 is selected.

Field Name	Bits	Type	Reset Value	Description
ddrc_reg_rd_mrr_data	31:0	ro	0x0	Mode register read Data. Valid when ddrc_co_rd_mrr_data_valid is high. Bits[7:0] carry the 8-bit MRR value. Valid for LPDDR2 only.

Register ([ddrc](#)) lpddr_ctrl0

Name	lpddr_ctrl0
Relative Address	0x000002A8
Absolute Address	0xF80062A8
Width	12 bits
Access Type	rw
Reset Value	0x00000000
Description	LPDDR2 Control 0

Register lpddr_ctrl0 Details

This registers is applicable only when LPDDR2 is selected.

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_mr4_margin	11:4	rw	0x0	UNUSED
reserved	3	rw	0x0	Reserved. Datasheet does not mention this field

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_derate_enable	2	rw	0x0	0: Timing parameter derating is disabled. 1: Timing parameter derating is enabled using MR4 read value.
reserved	1	rw	0x0	Reserved. Do not modify.
reg_ddrc_lpddr2	0	rw	0x0	0: DDR2 or DDR3 in use. 1: LPDDR2 in Use.

Register ([ddrc](#)) lpddr_ctrl1

Name	lpddr_ctrl1
Relative Address	0x000002AC
Absolute Address	0xF80062AC
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	LPDDR2 Control 1

Register lpddr_ctrl1 Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_mr4_read_interval	31:0	rw	0x0	Interval between two MR4 reads, USED to derate the timing parameters.

Register ([ddrc](#)) lpddr_ctrl2

Name	lpddr_ctrl2
Relative Address	0x000002B0
Absolute Address	0xF80062B0
Width	22 bits
Access Type	rw
Reset Value	0x003C0015
Description	LPDDR2 Control 2

Register lpddr_ctrl2 Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_t_mrw	21:12	rw	0x3C0	Time to wait during load mode register writes. Present only in designs configured to support LPDDR2. LPDDR2 typically requires value of 5.
reg_ddrc_idle_after_reset_x32	11:4	rw	0x1	Idle time after the reset command, tINIT4. Units: 32 clock cycles.
reg_ddrc_min_stable_clock_x1	3:0	rw	0x5	Time to wait after the first CKE high, tINIT2. Units: 1 clock cycle. LPDDR2 typically requires 5 x tCK delay.

Register ([ddrc](#)) lpddr_ctrl3

Name	lpddr_ctrl3
Relative Address	0x000002B4
Absolute Address	0xF80062B4
Width	18 bits
Access Type	rw
Reset Value	0x00000601
Description	LPDDR2 Control 3

Register lpddr_ctrl3 Details

Field Name	Bits	Type	Reset Value	Description
reg_ddrc_dev_zqinit_x32	17:8	rw	0x6	ZQ initial calibration, tZQINIT. Units: 32 clock cycles. LPDDR2 typically requires 1 us.
reg_ddrc_max_auto_init_x1024	7:0	rw	0x1	Maximum duration of the auto initialization, tINIT5. Units: 1024 clock cycles. LPDDR2 typically requires 10 us.

Register ([ddrc](#)) phy_wr_lvl_fsm

Name	phy_wr_lvl_fsm
Relative Address	0x000002B8
Absolute Address	0xF80062B8
Width	15 bits
Access Type	ro
Reset Value	0x00004444

Description PHY write leveling state machine status, read

Register phy_wr_lvl_fsm Details

Field Name	Bits	Type	Reset Value	Description
phy_reg_wr_level_fsm_slice3	14:12	ro	0x4	Write Leveling State information from Slice 3.
reserved	11	ro	0x0	Reserved
phy_reg_wr_level_fsm_slice2	10:8	ro	0x4	Write Leveling State information from Slice 2.
reserved	7	ro	0x0	Reserved
phy_reg_wr_level_fsm_slice1	6:4	ro	0x4	Write Leveling State information from Slice 1.
reserved	3	ro	0x0	Reserved
phy_reg_wr_level_fsm_slice0	2:0	ro	0x4	Write Leveling State information from Slice 0.

Register ([ddrc](#)) phy_rd_lvl_fsm

Name phy_rd_lvl_fsm

Relative Address 0x000002BC

Absolute Address 0xF80062BC

Width 16 bits

Access Type ro

Reset Value 0x00008888

Description PHY read leveling state machine status

Register phy_rd_lvl_fsm Details

Field Name	Bits	Type	Reset Value	Description
phy_reg_rd_level_fsm_slice3	15:12	ro	0x8	Read Leveling State information from Slice 3.
phy_reg_rd_level_fsm_slice2	11:8	ro	0x8	Read Leveling State information from Slice 2.
phy_reg_rd_level_fsm_slice1	7:4	ro	0x8	Read Leveling State information from Slice 1.
phy_reg_rd_level_fsm_slice0	3:0	ro	0x8	Read Leveling State information from Slice 0.

Register ([ddrc](#)) phy_gate_lvl_fsm

Name	phy_gate_lvl_fsm
Relative Address	0x000002C0
Absolute Address	0xF80062C0
Width	15 bits
Access Type	ro
Reset Value	0x00004444
Description	PHY gate leveling state machine status

Register phy_gate_lvl_fsm Details

Field Name	Bits	Type	Reset Value	Description
phy_reg_gate_level_fsm_slice3	14:12	ro	0x4	Gate Leveling State information from Slice 3.
reserved	11	ro	0x0	Reserved
phy_reg_gate_level_fsm_slice2	10:8	ro	0x4	Gate Leveling State information from Slice 2.
reserved	7	ro	0x0	Reserved
phy_reg_gate_level_fsm_slice1	6:4	ro	0x4	Gate Leveling State information from Slice 1.
reserved	3	ro	0x0	Reserved
phy_reg_gate_level_fsm_slice0	2:0	ro	0x4	Gate Leveling State information from Slice 0.

B.7 CoreSight Cross Trigger Interface (cti)

Module Name	CoreSight Cross Trigger Interface (cti)
Base Address	0xF8898000 debug_cpu_cti0 0xF8899000 debug_cpu_cti1 0xF8802000 debug_cti_etb_tpiu 0xF8809000 debug_cti_ftm
Description	Cross Trigger Interface This cti instance is for CPU 0.
Version	0.65
Doc Version	
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
CTICONTROL	0x00000000	1	rw	0x00000000	CTI Control Register
CTIINTACK	0x00000010	8	wo	0x00000000	CTI Interrupt Acknowledge Register
CTIAPPSET	0x00000014	4	rw	0x00000000	CTI Application Trigger Set Register
CTIAPPCLEAR	0x00000018	4	wo	0x00000000	CTI Application Trigger Clear Register
CTIAPPPULSE	0x0000001C	4	wo	0x00000000	CTI Application Pulse Register
CTIINEN0	0x00000020	4	rw	0x00000000	CTI Trigger to Channel Enable 0 Register
CTIINEN1	0x00000024	4	rw	0x00000000	CTI Trigger to Channel Enable 1 Register
CTIINEN2	0x00000028	4	rw	0x00000000	CTI Trigger to Channel Enable 2 Register
CTIINEN3	0x0000002C	4	rw	0x00000000	CTI Trigger to Channel Enable 3 Register
CTIINEN4	0x00000030	4	rw	0x00000000	CTI Trigger to Channel Enable 4 Register
CTIINEN5	0x00000034	4	rw	0x00000000	CTI Trigger to Channel Enable 5 Register
CTIINEN6	0x00000038	4	rw	0x00000000	CTI Trigger to Channel Enable 6 Register

Register Name	Address	Width	Type	Reset Value	Description
CTIINEN7	0x0000003C	4	rw	0x00000000	CTI Trigger to Channel Enable 7 Register
CTIOUTEN0	0x000000A0	4	rw	0x00000000	CTI Channel to Trigger Enable 0 Register
CTIOUTEN1	0x000000A4	4	rw	0x00000000	CTI Channel to Trigger Enable 1 Register
CTIOUTEN2	0x000000A8	4	rw	0x00000000	CTI Channel to Trigger Enable 2 Register
CTIOUTEN3	0x000000AC	4	rw	0x00000000	CTI Channel to Trigger Enable 3 Register
CTIOUTEN4	0x000000B0	4	rw	0x00000000	CTI Channel to Trigger Enable 4 Register
CTIOUTEN5	0x000000B4	4	rw	0x00000000	CTI Channel to Trigger Enable 5 Register
CTIOUTEN6	0x000000B8	4	rw	0x00000000	CTI Channel to Trigger Enable 6 Register
CTIOUTEN7	0x000000BC	4	rw	0x00000000	CTI Channel to Trigger Enable 7 Register
CTITRIGINSTATUS	0x00000130	8	ro	x	CTI Trigger In Status Register
CTITRIGOUTSTATUS	0x00000134	8	ro	0x00000000	CTI Trigger Out Status Register
CTICHINSTATUS	0x00000138	4	ro	x	CTI Channel In Status Register
CTICHOUTSTATUS	0x0000013C	4	ro	0x00000000	CTI Channel Out Status Register
CTIGATE	0x00000140	4	rw	0x0000000F	Enable CTI Channel Gate Register
ASICCTL	0x00000144	8	rw	0x00000000	External Multiplexor Control Register
ITCHINACK	0x00000EDC	4	wo	0x00000000	ITCHINACK Register
ITTRIGINACK	0x00000EE0	8	wo	0x00000000	ITTRIGINACK Register
ITCHOUT	0x00000EE4	4	wo	0x00000000	ITCHOUT Register
ITTRIGOUT	0x00000EE8	8	wo	0x00000000	ITTRIGOUT Register
ITCHOUTACK	0x00000EEC	4	ro	0x00000000	ITCHOUTACK Register
ITTRIGOUTACK	0x00000EF0	8	ro	0x00000000	ITTRIGOUTACK Register
ITCHIN	0x00000EF4	4	ro	0x00000000	ITCHIN Register
ITTRIGIN	0x00000EF8	8	ro	0x00000000	ITTRIGIN Register
ITCTRL	0x00000F00	1	rw	0x00000000	IT Control Register
CTSR	0x00000FA0	4	rw	0x0000000F	Claim Tag Set Register
CTCR	0x00000FA4	4	rw	0x00000000	Claim Tag Clear Register
LAR	0x00000FB0	32	wo	0x00000000	Lock Access Register

Register Name	Address	Width	Type	Reset Value	Description
LSR	0x00000FB4	3	ro	0x00000003	Lock Status Register
ASR	0x00000FB8	4	ro	x	Authentication Status Register
DEVID	0x00000FC8	20	ro	0x00040800	Device ID
DTIR	0x00000FCC	8	ro	0x00000014	Device Type Identifier Register
PERIPHID4	0x00000FD0	8	ro	0x00000004	Peripheral ID4
PERIPHID5	0x00000FD4	8	ro	0x00000000	Peripheral ID5
PERIPHID6	0x00000FD8	8	ro	0x00000000	Peripheral ID6
PERIPHID7	0x00000FDC	8	ro	0x00000000	Peripheral ID7
PERIPHID0	0x00000FE0	8	ro	0x00000006	Peripheral ID0
PERIPHID1	0x00000FE4	8	ro	0x000000B9	Peripheral ID1
PERIPHID2	0x00000FE8	8	ro	0x0000002B	Peripheral ID2
PERIPHID3	0x00000FEC	8	ro	0x00000000	Peripheral ID3
COMPID0	0x00000FF0	8	ro	0x0000000D	Component ID0
COMPID1	0x00000FF4	8	ro	0x00000090	Component ID1
COMPID2	0x00000FF8	8	ro	0x00000005	Component ID2
COMPID3	0x00000FFC	8	ro	0x000000B1	Component ID3

Register ([cti](#)) CTICONTROL

Name	CTICONTROL
Relative Address	0x00000000
Absolute Address	debug_cpu_cti0: 0xF8898000 debug_cpu_cti1: 0xF8899000 debug_cti_etb_tpiu: 0xF8802000 debug_cti_ftm: 0xF8809000
Width	1 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Control Register

Register CTICONTROL Details

Field Name	Bits	Type	Reset Value	Description
GLBEN	0	rw	0x0	Enables or disables the ECT. When disabled, all cross triggering mapping logic functionality is disabled for this processor.

Register ([cti](#)) CTIINTACK

Name	CTIINTACK
Relative Address	0x00000010
Absolute Address	debug_cpu_cti0: 0xF8898010 debug_cpu_cti1: 0xF8899010 debug_cti_etb_tpiu: 0xF8802010 debug_cti_ftm: 0xF8809010
Width	8 bits
Access Type	wo
Reset Value	0x00000000
Description	CTI Interrupt Acknowledge Register

Register CTIINTACK Details

Field Name	Bits	Type	Reset Value	Description
INTACK	7:0	wo	0x0	Acknowledges the corresponding CTITRIGOUT output: 1 = CTITRIGOUT is acknowledged and is cleared when MAPTRIGOUT is LOW. 0 = no effect There is one bit of the register for each CTITRIGOUT output.

Register ([cti](#)) CTIAPPSET

Name	CTIAPPSET
Relative Address	0x00000014
Absolute Address	debug_cpu_cti0: 0xF8898014 debug_cpu_cti1: 0xF8899014 debug_cti_etb_tpiu: 0xF8802014 debug_cti_ftm: 0xF8809014
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Application Trigger Set Register

Register CTIAPPSET Details

Field Name	Bits	Type	Reset Value	Description
APPSET	3:0	rw	0x0	Setting a bit HIGH generates a channel event for the selected channel. Read: 0 = application trigger inactive (reset) 1 = application trigger active. Write: 0 = no effect 1 = generate channel event. There is one bit of the register for each channel.

Register (cti) CTIAPPCLEAR

Name	CTIAPPCLEAR
Relative Address	0x00000018
Absolute Address	debug_cpu_cti0: 0xF8898018 debug_cpu_cti1: 0xF8899018 debug_cti_etb_tpiu: 0xF8802018 debug_cti_ftm: 0xF8809018
Width	4 bits
Access Type	wo
Reset Value	0x00000000
Description	CTI Application Trigger Clear Register

Register CTIAPPCLEAR Details

Field Name	Bits	Type	Reset Value	Description
APPCLEAR	3:0	wo	0x0	Clears corresponding bits in the CTIAPPSET register. 1 = application trigger disabled in the CTIAPPSET register 0 = no effect. There is one bit of the register for each channel.

Register (cti) CTIAPPPULSE

Name	CTIAPPPULSE
Relative Address	0x0000001C

Absolute Address	debug_cpu_cti0: 0xF889801C debug_cpu_cti1: 0xF889901C debug_cti_etb_tpiu: 0xF880201C debug_cti_ftm: 0xF880901C
Width	4 bits
Access Type	wo
Reset Value	0x00000000
Description	CTI Application Pulse Register

Register CTIAPPPULSE Details

Field Name	Bits	Type	Reset Value	Description
APPULSE	3:0	wo	0x0	Setting a bit HIGH generates a channel event pulse for the selected channel. Write: 1 = channel event pulse generated for one CTICLK period 0 = no effect. There is one bit of the register for each channel.

Register ([cti](#)) CTIINEN0

Name	CTIINEN0
Relative Address	0x00000020
Absolute Address	debug_cpu_cti0: 0xF8898020 debug_cpu_cti1: 0xF8899020 debug_cti_etb_tpiu: 0xF8802020 debug_cti_ftm: 0xF8809020
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Trigger to Channel Enable 0 Register

Register CTIINEN0 Details

Field Name	Bits	Type	Reset Value	Description
TRIGINEN	3:0	rw	0x0	<p>Enables a cross trigger event to the corresponding channel when an CTITRIGIN is activated.</p> <p>1 = enables the CTITRIGIN signal to generate an event on the respective channel of the CTM.</p> <p>There is one bit of the register for each of the four channels. For example in register CTIINEN0, TRIGINEN[0] set to 1 enables CTITRIGIN onto channel 0.</p> <p>0 = disables the CTITRIGIN signal from generating an event on the respective channel of the CTM.</p>

Register ([cti](#)) CTIINEN1

Name	CTIINEN1
Relative Address	0x00000024
Absolute Address	debug_cpu_cti0: 0xF8898024 debug_cpu_cti1: 0xF8899024 debug_cti_etb_tpiu: 0xF8802024 debug_cti_ftm: 0xF8809024
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Trigger to Channel Enable 1 Register

Register CTIINEN1 Details

Field Name	Bits	Type	Reset Value	Description
TRIGINEN	3:0	rw	0x0	<p>Enables a cross trigger event to the corresponding channel when an CTITRIGIN is activated.</p> <p>1 = enables the CTITRIGIN signal to generate an event on the respective channel of the CTM.</p> <p>There is one bit of the register for each of the four channels. For example in register CTIINEN0, TRIGINEN[0] set to 1 enables CTITRIGIN onto channel 0.</p> <p>0 = disables the CTITRIGIN signal from generating an event on the respective channel of the CTM.</p>

Register ([cti](#)) CTIINEN2

Name	CTIINEN2
Relative Address	0x00000028
Absolute Address	debug_cpu_cti0: 0xF8898028 debug_cpu_cti1: 0xF8899028 debug_cti_etb_tpiu: 0xF8802028 debug_cti_ftm: 0xF8809028
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Trigger to Channel Enable 2 Register

Register CTIINEN2 Details

Field Name	Bits	Type	Reset Value	Description
TRIGINEN	3:0	rw	0x0	Enables a cross trigger event to the corresponding channel when an CTITRIGIN is activated. 1 = enables the CTITRIGIN signal to generate an event on the respective channel of the CTM. There is one bit of the register for each of the four channels. For example in register CTIINEN0, TRIGINEN[0] set to 1 enables CTITRIGIN onto channel 0. 0 = disables the CTITRIGIN signal from generating an event on the respective channel of the CTM.

Register ([cti](#)) CTIINEN3

Name	CTIINEN3
Relative Address	0x0000002C
Absolute Address	debug_cpu_cti0: 0xF889802C debug_cpu_cti1: 0xF889902C debug_cti_etb_tpiu: 0xF880202C debug_cti_ftm: 0xF880902C
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Trigger to Channel Enable 3 Register

Register CTIINEN3 Details

Field Name	Bits	Type	Reset Value	Description
TRIGINEN	3:0	rw	0x0	<p>Enables a cross trigger event to the corresponding channel when an CTITRIGIN is activated.</p> <p>1 = enables the CTITRIGIN signal to generate an event on the respective channel of the CTM.</p> <p>There is one bit of the register for each of the four channels. For example in register CTIINEN0, TRIGINEN[0] set to 1 enables CTITRIGIN onto channel 0.</p> <p>0 = disables the CTITRIGIN signal from generating an event on the respective channel of the CTM.</p>

Register ([cti](#)) CTIINEN4

Name	CTIINEN4
Relative Address	0x00000030
Absolute Address	debug_cpu_cti0: 0xF8898030 debug_cpu_cti1: 0xF8899030 debug_cti_etb_tpiu: 0xF8802030 debug_cti_ftm: 0xF8809030
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Trigger to Channel Enable 4 Register

Register CTIINEN4 Details

Field Name	Bits	Type	Reset Value	Description
TRIGINEN	3:0	rw	0x0	<p>Enables a cross trigger event to the corresponding channel when an CTITRIGIN is activated.</p> <p>1 = enables the CTITRIGIN signal to generate an event on the respective channel of the CTM.</p> <p>There is one bit of the register for each of the four channels. For example in register CTIINEN0, TRIGINEN[0] set to 1 enables CTITRIGIN onto channel 0.</p> <p>0 = disables the CTITRIGIN signal from generating an event on the respective channel of the CTM.</p>

Register (cti) CTIINEN5

Name	CTIINEN5
Relative Address	0x00000034
Absolute Address	debug_cpu_cti0: 0xF8898034 debug_cpu_cti1: 0xF8899034 debug_cti_etb_tpiu: 0xF8802034 debug_cti_ftm: 0xF8809034
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Trigger to Channel Enable 5 Register

Register CTIINEN5 Details

Field Name	Bits	Type	Reset Value	Description
TRIGINEN	3:0	rw	0x0	Enables a cross trigger event to the corresponding channel when an CTITRIGIN is activated. 1 = enables the CTITRIGIN signal to generate an event on the respective channel of the CTM. There is one bit of the register for each of the four channels. For example in register CTIINEN0, TRIGINEN[0] set to 1 enables CTITRIGIN onto channel 0. 0 = disables the CTITRIGIN signal from generating an event on the respective channel of the CTM.

Register (cti) CTIINEN6

Name	CTIINEN6
Relative Address	0x00000038
Absolute Address	debug_cpu_cti0: 0xF8898038 debug_cpu_cti1: 0xF8899038 debug_cti_etb_tpiu: 0xF8802038 debug_cti_ftm: 0xF8809038
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Trigger to Channel Enable 6 Register

Register CTIINEN6 Details

Field Name	Bits	Type	Reset Value	Description
TRIGINEN	3:0	rw	0x0	<p>Enables a cross trigger event to the corresponding channel when an CTITRIGIN is activated.</p> <p>1 = enables the CTITRIGIN signal to generate an event on the respective channel of the CTM.</p> <p>There is one bit of the register for each of the four channels. For example in register CTIINEN0, TRIGINEN[0] set to 1 enables CTITRIGIN onto channel 0.</p> <p>0 = disables the CTITRIGIN signal from generating an event on the respective channel of the CTM.</p>

Register ([cti](#)) CTIINEN7

Name	CTIINEN7
Relative Address	0x0000003C
Absolute Address	debug_cpu_cti0: 0xF889803C debug_cpu_cti1: 0xF889903C debug_cti_etb_tpiu: 0xF880203C debug_cti_ftm: 0xF880903C
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Trigger to Channel Enable 7 Register

Register CTIINEN7 Details

Field Name	Bits	Type	Reset Value	Description
TRIGINEN	3:0	rw	0x0	<p>Enables a cross trigger event to the corresponding channel when an CTITRIGIN is activated.</p> <p>1 = enables the CTITRIGIN signal to generate an event on the respective channel of the CTM.</p> <p>There is one bit of the register for each of the four channels. For example in register CTIINEN0, TRIGINEN[0] set to 1 enables CTITRIGIN onto channel 0.</p> <p>0 = disables the CTITRIGIN signal from generating an event on the respective channel of the CTM.</p>

Register ([cti](#)) CTIOUTEN0

Name	CTIOUTEN0
Relative Address	0x000000A0
Absolute Address	debug_cpu_cti0: 0xF88980A0 debug_cpu_cti1: 0xF88990A0 debug_cti_etb_tpiu: 0xF88020A0 debug_cti_ftm: 0xF88090A0
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Channel to Trigger Enable 0 Register

Register CTIOUTEN0 Details

Field Name	Bits	Type	Reset Value	Description
TRIGOUTEN	3:0	rw	0x0	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate an CTITRIGOUT output: 0 = the channel input (CTICHIN) from the CTM is not routed to the CTITRIGOUT output 1 = the channel input (CTICHIN) from the CTM is routed to the CTITRIGOUT output. There is one bit for each of the four channels. For example in register CTIOUTEN0, enabling bit 0 enables CTICHIN[0] to cause a trigger event on the CTITRIGOUT[0] output.

Register ([cti](#)) CTIOUTEN1

Name	CTIOUTEN1
Relative Address	0x000000A4
Absolute Address	debug_cpu_cti0: 0xF88980A4 debug_cpu_cti1: 0xF88990A4 debug_cti_etb_tpiu: 0xF88020A4 debug_cti_ftm: 0xF88090A4
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Channel to Trigger Enable 1 Register

Register CTIOUTEN1 Details

Field Name	Bits	Type	Reset Value	Description
TRIGOUTEN	3:0	rw	0x0	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate an CTITRIGOUT output: 0 = the channel input (CTICHIN) from the CTM is not routed to the CTITRIGOUT output 1 = the channel input (CTICHIN) from the CTM is routed to the CTITRIGOUT output. There is one bit for each of the four channels. For example in register CTIOUTEN0, enabling bit 0 enables CTICHIN[0] to cause a trigger event on the CTITRIGOUT[0] output.

Register ([cti](#)) CTIOUTEN2

Name	CTIOUTEN2
Relative Address	0x000000A8
Absolute Address	debug_cpu_cti0: 0xF88980A8 debug_cpu_cti1: 0xF88990A8 debug_cti_etb_tpiu: 0xF88020A8 debug_cti_ftm: 0xF88090A8
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Channel to Trigger Enable 2 Register

Register CTIOUTEN2 Details

Field Name	Bits	Type	Reset Value	Description
TRIGOUTEN	3:0	rw	0x0	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate an CTITRIGOUT output: 0 = the channel input (CTICHIN) from the CTM is not routed to the CTITRIGOUT output 1 = the channel input (CTICHIN) from the CTM is routed to the CTITRIGOUT output. There is one bit for each of the four channels. For example in register CTIOUTEN0, enabling bit 0 enables CTICHIN[0] to cause a trigger event on the CTITRIGOUT[0] output.

Register (cti) CTIOUTEN3

Name	CTIOUTEN3
Relative Address	0x000000AC
Absolute Address	debug_cpu_cti0: 0xF88980AC debug_cpu_cti1: 0xF88990AC debug_cti_etb_tpiu: 0xF88020AC debug_cti_ftm: 0xF88090AC
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Channel to Trigger Enable 3 Register

Register CTIOUTEN3 Details

Field Name	Bits	Type	Reset Value	Description
TRIGOUTEN	3:0	rw	0x0	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate an CTITRIGOUT output: 0 = the channel input (CTICHIN) from the CTM is not routed to the CTITRIGOUT output 1 = the channel input (CTICHIN) from the CTM is routed to the CTITRIGOUT output. There is one bit for each of the four channels. For example in register CTIOUTEN0, enabling bit 0 enables CTICHIN[0] to cause a trigger event on the CTITRIGOUT[0] output.

Register (cti) CTIOUTEN4

Name	CTIOUTEN4
Relative Address	0x000000B0
Absolute Address	debug_cpu_cti0: 0xF88980B0 debug_cpu_cti1: 0xF88990B0 debug_cti_etb_tpiu: 0xF88020B0 debug_cti_ftm: 0xF88090B0
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Channel to Trigger Enable 4 Register

Register CTIOUTEN4 Details

Field Name	Bits	Type	Reset Value	Description
TRIGOUTEN	3:0	rw	0x0	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate an CTITRIGOUT output: 0 = the channel input (CTICHIN) from the CTM is not routed to the CTITRIGOUT output 1 = the channel input (CTICHIN) from the CTM is routed to the CTITRIGOUT output. There is one bit for each of the four channels. For example in register CTIOUTEN0, enabling bit 0 enables CTICHIN[0] to cause a trigger event on the CTITRIGOUT[0] output.

Register ([cti](#)) CTIOUTEN5

Name	CTIOUTEN5
Relative Address	0x000000B4
Absolute Address	debug_cpu_cti0: 0xF88980B4 debug_cpu_cti1: 0xF88990B4 debug_cti_etb_tpiu: 0xF88020B4 debug_cti_ftm: 0xF88090B4
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Channel to Trigger Enable 5 Register

Register CTIOUTEN5 Details

Field Name	Bits	Type	Reset Value	Description
TRIGOUTEN	3:0	rw	0x0	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate an CTITRIGOUT output: 0 = the channel input (CTICHIN) from the CTM is not routed to the CTITRIGOUT output 1 = the channel input (CTICHIN) from the CTM is routed to the CTITRIGOUT output. There is one bit for each of the four channels. For example in register CTIOUTEN0, enabling bit 0 enables CTICHIN[0] to cause a trigger event on the CTITRIGOUT[0] output.

Register (cti) CTIOUTEN6

Name	CTIOUTEN6
Relative Address	0x000000B8
Absolute Address	debug_cpu_cti0: 0xF88980B8 debug_cpu_cti1: 0xF88990B8 debug_cti_etb_tpiu: 0xF88020B8 debug_cti_ftm: 0xF88090B8
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Channel to Trigger Enable 6 Register

Register CTIOUTEN6 Details

Field Name	Bits	Type	Reset Value	Description
TRIGOUTEN	3:0	rw	0x0	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate an CTITRIGOUT output: 0 = the channel input (CTICHIN) from the CTM is not routed to the CTITRIGOUT output 1 = the channel input (CTICHIN) from the CTM is routed to the CTITRIGOUT output. There is one bit for each of the four channels. For example in register CTIOUTEN0, enabling bit 0 enables CTICHIN[0] to cause a trigger event on the CTITRIGOUT[0] output.

Register (cti) CTIOUTEN7

Name	CTIOUTEN7
Relative Address	0x000000BC
Absolute Address	debug_cpu_cti0: 0xF88980BC debug_cpu_cti1: 0xF88990BC debug_cti_etb_tpiu: 0xF88020BC debug_cti_ftm: 0xF88090BC
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	CTI Channel to Trigger Enable 7 Register

Register CTIOUTEN7 Details

Field Name	Bits	Type	Reset Value	Description
TRIGOUTEN	3:0	rw	0x0	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate an CTITRIGOUT output: 0 = the channel input (CTICHIN) from the CTM is not routed to the CTITRIGOUT output 1 = the channel input (CTICHIN) from the CTM is routed to the CTITRIGOUT output. There is one bit for each of the four channels. For example in register CTIOUTEN0, enabling bit 0 enables CTICHIN[0] to cause a trigger event on the CTITRIGOUT[0] output.

Register ([cti](#)) CTITRIGINSTATUS

Name	CTITRIGINSTATUS
Relative Address	0x00000130
Absolute Address	debug_cpu_cti0: 0xF8898130 debug_cpu_cti1: 0xF8899130 debug_cti_etb_tpiu: 0xF8802130 debug_cti_ftm: 0xF8809130
Width	8 bits
Access Type	ro
Reset Value	x
Description	CTI Trigger In Status Register

Register CTITRIGINSTATUS Details

Field Name	Bits	Type	Reset Value	Description
TRIGINSTATUS	7:0	ro	x	Shows the status of the CTITRIGIN inputs: 1 = CTITRIGIN is active 0 = CTITRIGIN is inactive. Because the register provides a view of the raw CTITRIGIN inputs, the reset value is unknown. There is one bit of the register for each trigger input.

Register ([cti](#)) CTITRIGOUTSTATUS

Name	CTITRIGOUTSTATUS
Relative Address	0x00000134

Absolute Address	debug_cpu_cti0: 0xF8898134 debug_cpu_cti1: 0xF8899134 debug_cti_etb_tpiu: 0xF8802134 debug_cti_ftm: 0xF8809134
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	CTI Trigger Out Status Register

Register CTITRIGOUTSTATUS Details

Field Name	Bits	Type	Reset Value	Description
TRIGOUTSTATUS	7:0	ro	0x0	Shows the status of the CTITRIGOUT outputs. 1 = CTITRIGOUT is active 0 = CTITRIGOUT is inactive (reset). There is one bit of the register for each trigger output.

Register ([cti](#)) CTICHINSTATUS

Name	CTICHINSTATUS
Relative Address	0x00000138
Absolute Address	debug_cpu_cti0: 0xF8898138 debug_cpu_cti1: 0xF8899138 debug_cti_etb_tpiu: 0xF8802138 debug_cti_ftm: 0xF8809138
Width	4 bits
Access Type	ro
Reset Value	x
Description	CTI Channel In Status Register

Register CTICHINSTATUS Details

Field Name	Bits	Type	Reset Value	Description
CTICHINSTATUS	3:0	ro	x	Shows the status of the CTICHIN inputs: 1 = CTICHIN is active 0 = CTICHIN is inactive. Because the register provides a view of the raw CTICHIN inputs from the CTM, the reset value is unknown. There is one bit of the register for each channel input.

Register ([cti](#)) CTICHOUTSTATUS

Name	CTICHOUTSTATUS
Relative Address	0x0000013C
Absolute Address	debug_cpu_cti0: 0xF889813C debug_cpu_cti1: 0xF889913C debug_cti_etb_tpiu: 0xF880213C debug_cti_ftm: 0xF880913C
Width	4 bits
Access Type	ro
Reset Value	0x00000000
Description	CTI Channel Out Status Register

Register CTICHOUTSTATUS Details

Field Name	Bits	Type	Reset Value	Description
CTICHOUTSTATUS	3:0	ro	0x0	Shows the status of the CTICHOUT outputs. 1 = CTICHOUT is active 0 = CTICHOUT is inactive (reset). There is one bit of the register for each channel output.

Register ([cti](#)) CTIGATE

Name	CTIGATE
Relative Address	0x00000140
Absolute Address	debug_cpu_cti0: 0xF8898140 debug_cpu_cti1: 0xF8899140 debug_cti_etb_tpiu: 0xF8802140 debug_cti_ftm: 0xF8809140
Width	4 bits
Access Type	rw
Reset Value	0x0000000F
Description	Enable CTI Channel Gate Register

Register CTIGATE Details

Field Name	Bits	Type	Reset Value	Description
CTIGATEEN3	3	rw	0x1	Enable CTICHOUT3.
CTIGATEEN2	2	rw	0x1	Enable CTICHOUT2.

Field Name	Bits	Type	Reset Value	Description
CTIGATEEN1	1	rw	0x1	Enable CTICHOUT1.
CTIGATEEN0	0	rw	0x1	Enable CTICHOUT0.

Register (cti) ASICCTL

Name	ASICCTL
Relative Address	0x00000144
Absolute Address	debug_cpu_cti0: 0xF8898144 debug_cpu_cti1: 0xF8899144 debug_cti_etb_tpiu: 0xF8802144 debug_cti_ftm: 0xF8809144
Width	8 bits
Access Type	rw
Reset Value	0x00000000
Description	External Multiplexor Control Register

Register ASICCTL Details

Field Name	Bits	Type	Reset Value	Description
ASICCTL	7:0	rw	0x0	Implementation defined ASIC control, value written to the register is output on ASICCTL[7:0].

Register (cti) ITCHINACK

Name	ITCHINACK
Relative Address	0x00000EDC
Absolute Address	debug_cpu_cti0: 0xF8898EDC debug_cpu_cti1: 0xF8899EDC debug_cti_etb_tpiu: 0xF8802EDC debug_cti_ftm: 0xF8809EDC
Width	4 bits
Access Type	wo
Reset Value	0x00000000
Description	ITCHINACK Register

Register ITCHINACK Details

Field Name	Bits	Type	Reset Value	Description
CTCHINACK	3:0	wo	0x0	Set the value of the CTCHINACK outputs

Register (cti) ITTRIGINACK

Name	ITTRIGINACK
Relative Address	0x00000EE0
Absolute Address	debug_cpu_cti0: 0xF8898EE0 debug_cpu_cti1: 0xF8899EE0 debug_cti_etb_tpiu: 0xF8802EE0 debug_cti_ftm: 0xF8809EE0
Width	8 bits
Access Type	wo
Reset Value	0x00000000
Description	ITTRIGINACK Register

Register ITTRIGINACK Details

Field Name	Bits	Type	Reset Value	Description
CTTRIGINACK	7:0	wo	0x0	Set the value of the CTTRIGINACK outputs

Register (cti) ITCHOUT

Name	ITCHOUT
Relative Address	0x00000EE4
Absolute Address	debug_cpu_cti0: 0xF8898EE4 debug_cpu_cti1: 0xF8899EE4 debug_cti_etb_tpiu: 0xF8802EE4 debug_cti_ftm: 0xF8809EE4
Width	4 bits
Access Type	wo
Reset Value	0x00000000
Description	ITCHOUT Register

Register ITCHOUT Details

Field Name	Bits	Type	Reset Value	Description
CTCHOUT	3:0	wo	0x0	Set the value of the CTCHOUT outputs

Register (cti) ITTRIGOUT

Name	ITTRIGOUT
Relative Address	0x00000EE8

Absolute Address	debug_cpu_cti0: 0xF8898EE8 debug_cpu_cti1: 0xF8899EE8 debug_cti_etb_tpiu: 0xF8802EE8 debug_cti_ftm: 0xF8809EE8
Width	8 bits
Access Type	wo
Reset Value	0x00000000
Description	ITTRIGOUT Register

Register ITTRIGOUT Details

Field Name	Bits	Type	Reset Value	Description
CTTRIGOUT	7:0	wo	0x0	Set the value of the CTTRIGOUT outputs

Register ([cti](#)) ITCHOUTACK

Name	ITCHOUTACK
Relative Address	0x00000EEC
Absolute Address	debug_cpu_cti0: 0xF8898EEC debug_cpu_cti1: 0xF8899EEC debug_cti_etb_tpiu: 0xF8802EEC debug_cti_ftm: 0xF8809EEC
Width	4 bits
Access Type	ro
Reset Value	0x00000000
Description	ITCHOUTACK Register

Register ITCHOUTACK Details

Field Name	Bits	Type	Reset Value	Description
CTCHOUTACK	3:0	ro	0x0	Read the values of the CTCHOUTACK inputs

Register ([cti](#)) ITTRIGOUTACK

Name	ITTRIGOUTACK
Relative Address	0x00000EF0
Absolute Address	debug_cpu_cti0: 0xF8898EF0 debug_cpu_cti1: 0xF8899EF0 debug_cti_etb_tpiu: 0xF8802EF0 debug_cti_ftm: 0xF8809EF0

Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	ITTRIGOUTACK Register

Register ITTRIGOUTACK Details

Field Name	Bits	Type	Reset Value	Description
CTTRIGOUTACK	7:0	ro	0x0	Read the values of the CTTRIGOUTACK inputs

Register ([cti](#)) ITCHIN

Name	ITCHIN
Relative Address	0x00000EF4
Absolute Address	debug_cpu_cti0: 0xF8898EF4 debug_cpu_cti1: 0xF8899EF4 debug_cti_etb_tpiu: 0xF8802EF4 debug_cti_ftm: 0xF8809EF4
Width	4 bits
Access Type	ro
Reset Value	0x00000000
Description	ITCHIN Register

Register ITCHIN Details

Field Name	Bits	Type	Reset Value	Description
CTCHIN	3:0	ro	0x0	Read the values of the CTCHIN inputs

Register ([cti](#)) ITTRIGIN

Name	ITTRIGIN
Relative Address	0x00000EF8
Absolute Address	debug_cpu_cti0: 0xF8898EF8 debug_cpu_cti1: 0xF8899EF8 debug_cti_etb_tpiu: 0xF8802EF8 debug_cti_ftm: 0xF8809EF8
Width	8 bits
Access Type	ro
Reset Value	0x00000000

Description ITTRIGIN Register

Register ITTRIGIN Details

Field Name	Bits	Type	Reset Value	Description
CTTRIGIN	7:0	ro	0x0	Read the values of the CTTRIGIN inputs

Register ([cti](#)) ITCTRL

Name ITCTRL

Relative Address 0x00000F00

Absolute Address debug_cpu_cti0: 0xF8898F00
debug_cpu_cti1: 0xF8899F00
debug_cti_etb_tpiu: 0xF8802F00
debug_cti_ftm: 0xF8809F00

Width 1 bits

Access Type rw

Reset Value 0x00000000

Description IT Control Register

Register ITCTRL Details

Field Name	Bits	Type	Reset Value	Description
	0	rw	0x0	Enable IT Registers

Register ([cti](#)) CTSR

Name CTSR

Relative Address 0x00000FA0

Absolute Address debug_cpu_cti0: 0xF8898FA0
debug_cpu_cti1: 0xF8899FA0
debug_cti_etb_tpiu: 0xF8802FA0
debug_cti_ftm: 0xF8809FA0

Width 4 bits

Access Type rw

Reset Value 0x0000000F

Description Claim Tag Set Register

Register CTSR Details

Field Name	Bits	Type	Reset Value	Description
SET	3:0	rw	0xF	The claim tag register is used for any interrogating tools to determine if the device is being programmed or has been programmed. Read: 1= Claim tag is implemented, 0 = Claim tag is not implemented Write: 1= Set claim tag bit, 0= No effect

Register ([cti](#)) CTCR

Name	CTCR
Relative Address	0x00000FA4
Absolute Address	debug_cpu_cti0: 0xF8898FA4 debug_cpu_cti1: 0xF8899FA4 debug_cti_etb_tpiu: 0xF8802FA4 debug_cti_ftm: 0xF8809FA4
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	Claim Tag Clear Register

Register CTCR Details

Field Name	Bits	Type	Reset Value	Description
CLEAR	3:0	rw	0x0	The claim tag register is used for any interrogating tools to determine if the device is being programmed or has been programmed. Read: Current value of claim tag. Write: 1= Clear claim tag bit, 0= No effect

Register ([cti](#)) LAR

Name	LAR
Relative Address	0x00000FB0
Absolute Address	debug_cpu_cti0: 0xF8898FB0 debug_cpu_cti1: 0xF8899FB0 debug_cti_etb_tpiu: 0xF8802FB0 debug_cti_ftm: 0xF8809FB0

Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	Lock Access Register

Register LAR Details

Field Name	Bits	Type	Reset Value	Description
KEY	31:0	wo	0x0	<p>Write Access Code.</p> <p>Write behavior depends on PADDRDBG31 pin:</p> <ul style="list-style-type: none"> - PADDRDBG31=0 (lower 2GB): <p>After reset (via PRESETDBGn), CTI is locked, i.e., writes to all other registers using lower 2GB addresses are ignored.</p> <p>To unlock, 0xC5ACCE55 must be written this register.</p> <p>After the required registers are written, to lock again, write a value other than 0xC5ACCE55 to this register.</p> <ul style="list-style-type: none"> - PADDRDBG31=1 (upper 2GB): <p>CTI is unlocked when upper 2GB addresses are used to write to all the registers.</p> <p>However, write to this register is ignored using a upper 2GB address!</p> <p>Note: read from this register always returns 0, regardless of PADDRDBG31.</p>

Register ([cti](#)) LSR

Name	LSR
Relative Address	0x0000FB4
Absolute Address	debug_cpu_cti0: 0xF8898FB4 debug_cpu_cti1: 0xF8899FB4 debug_cti_etb_tpiu: 0xF8802FB4 debug_cti_ftm: 0xF8809FB4
Width	3 bits
Access Type	ro
Reset Value	0x00000003
Description	Lock Status Register

Register LSR Details

Field Name	Bits	Type	Reset Value	Description
8BIT	2	ro	0x0	Set to 0 since CTI implements a 32-bit lock access register
STATUS	1	ro	0x1	Read behavior depends on PADDRDBG31 pin: - PADDRDBG31=0 (lower 2GB): When a lower 2GB address is used to read this register, this bit indicates whether CTI is in locked state (1= locked, 0= unlocked). - PADDRDBG31=1 (upper 2GB): always returns 0.
IMP	0	ro	0x1	Read behavior depends on PADDRDBG31 pin: - PADDRDBG31=0 (lower 2GB): always returns 1, meaning lock mechanism are implemented. - PADDRDBG31=1 (upper 2GB): always returns 0, meaning lock mechanism is NOT implemented.

Register (cti) ASR

Name	ASR
Relative Address	0x00000FB8
Absolute Address	debug_cpu_cti0: 0xF8898FB8 debug_cpu_cti1: 0xF8899FB8 debug_cti_etb_tpiu: 0xF8802FB8 debug_cti_ftm: 0xF8809FB8
Width	4 bits
Access Type	ro
Reset Value	x
Description	Authentication Status Register

Register ASR Details

Field Name	Bits	Type	Reset Value	Description
NIDEN	3	ro	x	Current value of noninvasive debug enable signals
NIDEN_CTL	2	ro	0x1	Non-invasive debug controlled
IDEN	1	ro	x	Current value of invasive debug enable signals
IDEN_CTL	0	ro	0x1	Invasive debug controlled

Register (cti) DEVID

Name	DEVID
Relative Address	0x0000FC8
Absolute Address	debug_cpu_cti0: 0xF8898FC8 debug_cpu_cti1: 0xF8899FC8 debug_cti_etb_tpiu: 0xF8802FC8 debug_cti_ftm: 0xF8809FC8
Width	20 bits
Access Type	ro
Reset Value	0x00040800
Description	Device ID

Register DEVID Details

Field Name	Bits	Type	Reset Value	Description
NumChan	19:16	ro	0x4	Number of channels available
NumTrig	15:8	ro	0x8	Number of triggers available
res	7:5	ro	0x0	reserved
ExtMux	4:0	ro	0x0	no external muxing

Register (cti) DTIR

Name	DTIR
Relative Address	0x0000FCC
Absolute Address	debug_cpu_cti0: 0xF8898FCC debug_cpu_cti1: 0xF8899FCC debug_cti_etb_tpiu: 0xF8802FCC debug_cti_ftm: 0xF8809FCC
Width	8 bits
Access Type	ro
Reset Value	0x00000014
Description	Device Type Identifier Register

Register DTIR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x14	major type is a debug control logic component, sub-type is cross trigger

Register (cti) PERIPHID4

Name	PERIPHID4
Relative Address	0x0000FD0
Absolute Address	debug_cpu_cti0: 0xF8898FD0 debug_cpu_cti1: 0xF8899FD0 debug_cti_etb_tpiu: 0xF8802FD0 debug_cti_ftm: 0xF8809FD0
Width	8 bits
Access Type	ro
Reset Value	0x00000004
Description	Peripheral ID4

Register PERIPHID4 Details

Field Name	Bits	Type	Reset Value	Description
4KB_count	7:4	ro	0x0	4KB Count, set to 0
JEP106ID	3:0	ro	0x4	JEP106 continuation code

Register (cti) PERIPHID5

Name	PERIPHID5
Relative Address	0x0000FD4
Absolute Address	debug_cpu_cti0: 0xF8898FD4 debug_cpu_cti1: 0xF8899FD4 debug_cti_etb_tpiu: 0xF8802FD4 debug_cti_ftm: 0xF8809FD4
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID5

Register PERIPHID5 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register (cti) PERIPHID6

Name	PERIPHID6
------	-----------

Relative Address	0x00000FD8
Absolute Address	debug_cpu_cti0: 0xF8898FD8 debug_cpu_cti1: 0xF8899FD8 debug_cti_etb_tpiu: 0xF8802FD8 debug_cti_ftm: 0xF8809FD8
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID6

Register PERIPID6 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([cti](#)) PERIPID7

Name	PERIPID7
Relative Address	0x00000FDC
Absolute Address	debug_cpu_cti0: 0xF8898FDC debug_cpu_cti1: 0xF8899FDC debug_cti_etb_tpiu: 0xF8802FDC debug_cti_ftm: 0xF8809FDC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID7

Register PERIPID7 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([cti](#)) PERIPID0

Name	PERIPID0
Relative Address	0x00000FE0

Absolute Address	debug_cpu_cti0: 0xF8898FE0 debug_cpu_cti1: 0xF8899FE0 debug_cti_etb_tpiu: 0xF8802FE0 debug_cti_ftm: 0xF8809FE0
Width	8 bits
Access Type	ro
Reset Value	0x00000006
Description	Peripheral ID0

Register PERIPHID0 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x6	PartNumber0

Register ([cti](#)) PERIPHID1

Name	PERIPHID1
Relative Address	0x00000FE4
Absolute Address	debug_cpu_cti0: 0xF8898FE4 debug_cpu_cti1: 0xF8899FE4 debug_cti_etb_tpiu: 0xF8802FE4 debug_cti_ftm: 0xF8809FE4
Width	8 bits
Access Type	ro
Reset Value	0x000000B9
Description	Peripheral ID1

Register PERIPHID1 Details

Field Name	Bits	Type	Reset Value	Description
JEP106ID	7:4	ro	0xB	JEP106 Identity Code [3:0]
PartNumber1	3:0	ro	0x9	PartNumber1

Register ([cti](#)) PERIPHID2

Name	PERIPHID2
Relative Address	0x00000FE8

Absolute Address	debug_cpu_cti0: 0xF8898FE8 debug_cpu_cti1: 0xF8899FE8 debug_cti_etb_tpiu: 0xF8802FE8 debug_cti_ftm: 0xF8809FE8
Width	8 bits
Access Type	ro
Reset Value	0x0000002B
Description	Peripheral ID2

Register PERIPID2 Details

Field Name	Bits	Type	Reset Value	Description
RevNum	7:4	ro	0x2	Revision number of Peripheral
JEDEC	3	ro	0x1	Indicates that a JEDEC assigned value is used
JEP106ID	2:0	ro	0x3	JEP106 Identity Code [6:4]

Register (cti) PERIPID3

Name	PERIPID3
Relative Address	0x00000FEC
Absolute Address	debug_cpu_cti0: 0xF8898FEC debug_cpu_cti1: 0xF8899FEC debug_cti_etb_tpiu: 0xF8802FEC debug_cti_ftm: 0xF8809FEC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID3

Register PERIPID3 Details

Field Name	Bits	Type	Reset Value	Description
RevAnd	7:4	ro	0x0	RevAnd, at top level
CustMod	3:0	ro	0x0	Customer Modified

Register (cti) COMPID0

Name	COMPID0
Relative Address	0x00000FF0

Absolute Address	debug_cpu_cti0: 0xF8898FF0 debug_cpu_cti1: 0xF8899FF0 debug_cti_etb_tpiu: 0xF8802FF0 debug_cti_ftm: 0xF8809FF0
Width	8 bits
Access Type	ro
Reset Value	0x0000000D
Description	Component ID0

Register COMPID0 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0xD	Preamble

Register ([cti](#)) COMPID1

Name	COMPID1
Relative Address	0x00000FF4
Absolute Address	debug_cpu_cti0: 0xF8898FF4 debug_cpu_cti1: 0xF8899FF4 debug_cti_etb_tpiu: 0xF8802FF4 debug_cti_ftm: 0xF8809FF4
Width	8 bits
Access Type	ro
Reset Value	0x00000090
Description	Component ID1

Register COMPID1 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x90	Preamble

Register ([cti](#)) COMPID2

Name	COMPID2
Relative Address	0x00000FF8
Absolute Address	debug_cpu_cti0: 0xF8898FF8 debug_cpu_cti1: 0xF8899FF8 debug_cti_etb_tpiu: 0xF8802FF8 debug_cti_ftm: 0xF8809FF8

Width	8 bits
Access Type	ro
Reset Value	0x00000005
Description	Component ID2

Register COMPID2 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x5	Preamble

Register ([cti](#)) COMPID3

Name	COMPID3
Relative Address	0x0000FFC
Absolute Address	debug_cpu_cti0: 0xF8898FFC debug_cpu_cti1: 0xF8899FFC debug_cti_etb_tpiu: 0xF8802FFC debug_cti_ftm: 0xF8809FFC
Width	8 bits
Access Type	ro
Reset Value	0x000000B1
Description	Component ID3

Register COMPID3 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0xB1	Preamble

B.8 Performance Monitor Unit (cortexa9_pmu)

Module Name	Performance Monitor Unit (cortexa9_pmu)
Base Address	0xF8891000 debug_cpu_pmu0 0xF8893000 debug_cpu_pmu1
Description	Cortex A9 Performance Monitoring Unit This pmu instance is for CPU 0.
Version	1.0
Doc Version	1.0
Vendor Info	ARM

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
PMXEVCNTR0	0x00000000	32	rw	x	PMU event counter 0
PMXEVCNTR1	0x00000004	32	rw	x	PMU event counter 1
PMXEVCNTR2	0x00000008	32	rw	x	PMU event counter 2
PMXEVCNTR3	0x0000000C	32	rw	x	PMU event counter 3
PMXEVCNTR4	0x00000010	32	rw	x	PMU event counter 4
PMXEVCNTR5	0x00000014	32	rw	x	PMU event counter 5
PMCCNTR	0x0000007C	32	rw	x	pmccntr
PMXEVTYPER0	0x00000400	32	rw	x	pmevtyper0
PMXEVTYPER1	0x00000404	32	rw	x	pmevtyper1
PMXEVTYPER2	0x00000408	32	rw	x	pmevtyper2
PMXEVTYPER3	0x0000040C	32	rw	x	pmevtyper3
PMXEVTYPER4	0x00000410	32	rw	x	pmevtyper4
PMXEVTYPER5	0x00000414	32	rw	x	pmevtyper5
PMCNTENSET	0x00000C00	32	rw	0x00000000	pmcntenreset
PMCNTENCLR	0x00000C20	32	rw	0x00000000	pmcntenclr
PMINTENSET	0x00000C40	32	rw	0x00000000	pmintenreset
PMINTENCLR	0x00000C60	32	rw	0x00000000	pmintenclr
PMOVSr	0x00000C80	32	rw	x	pmovsr
PMSWINC	0x00000CA0	32	wo	x	pmswinc
PMCR	0x00000E04	32	rw	0x41093000	pmcr
PMUSERENR	0x00000E08	32	rw	0x00000000	pmuserenr

Register ([cortexa9_pmu](#)) PMXEVCNTR0

Name	PMXEVCNTR0
Relative Address	0x00000000
Absolute Address	debug_cpu_pmu0: 0xF8891000 debug_cpu_pmu1: 0xF8893000
Width	32 bits
Access Type	rw
Reset Value	x
Description	PMU event counter 0

Register PMXEVCNTR0 Details

Field Name	Bits	Type	Reset Value	Description
PMXEVCNTR0	31:0	rw	x	PMU event counter 0

Register ([cortexa9_pmu](#)) PMXEVCNTR1

Name	PMXEVCNTR1
Relative Address	0x00000004
Absolute Address	debug_cpu_pmu0: 0xF8891004 debug_cpu_pmu1: 0xF8893004
Width	32 bits
Access Type	rw
Reset Value	x
Description	PMU event counter 1

Register PMXEVCNTR1 Details

Field Name	Bits	Type	Reset Value	Description
PMXEVCNTR1	31:0	rw	x	PMU event counter 1

Register ([cortexa9_pmu](#)) PMXEVCNTR2

Name	PMXEVCNTR2
Relative Address	0x00000008
Absolute Address	debug_cpu_pmu0: 0xF8891008 debug_cpu_pmu1: 0xF8893008
Width	32 bits

Access Type	rw
Reset Value	x
Description	PMU event counter 2

Register PMXEVCNTR2 Details

Field Name	Bits	Type	Reset Value	Description
PMXEVCNTR2	31:0	rw	x	PMU event counter 2

Register ([cortexa9_pmu](#)) PMXEVCNTR3

Name	PMXEVCNTR3
Relative Address	0x0000000C
Absolute Address	debug_cpu_pmu0: 0xF889100C debug_cpu_pmu1: 0xF889300C
Width	32 bits
Access Type	rw
Reset Value	x
Description	PMU event counter 3

Register PMXEVCNTR3 Details

Field Name	Bits	Type	Reset Value	Description
PMXEVCNTR3	31:0	rw	x	PMU event counter 3

Register ([cortexa9_pmu](#)) PMXEVCNTR4

Name	PMXEVCNTR4
Relative Address	0x00000010
Absolute Address	debug_cpu_pmu0: 0xF8891010 debug_cpu_pmu1: 0xF8893010
Width	32 bits
Access Type	rw
Reset Value	x
Description	PMU event counter 4

Register PMXEVCNTR4 Details

Field Name	Bits	Type	Reset Value	Description
PMXEVCNTR4	31:0	rw	x	PMU event counter 4

Register ([cortexa9_pmu](#)) PMXEVCNTR5

Name	PMXEVCNTR5
Relative Address	0x00000014
Absolute Address	debug_cpu_pmu0: 0xF8891014 debug_cpu_pmu1: 0xF8893014
Width	32 bits
Access Type	rw
Reset Value	x
Description	PMU event counter 5

Register PMXEVCNTR5 Details

Field Name	Bits	Type	Reset Value	Description
PMXEVCNTR5	31:0	rw	x	PMU event counter 5

Register ([cortexa9_pmu](#)) PMCCNTR

Name	PMCCNTR
Relative Address	0x0000007C
Absolute Address	debug_cpu_pmu0: 0xF889107C debug_cpu_pmu1: 0xF889307C
Width	32 bits
Access Type	rw
Reset Value	x
Description	pmccntr

Register PMCCNTR Details

Field Name	Bits	Type	Reset Value	Description
PMCCNTR	31:0	rw	x	pmccntr

Register ([cortexa9_pmu](#)) PMXEVTYPER0

Name	PMXEVTYPER0
Relative Address	0x00000400
Absolute Address	debug_cpu_pmu0: 0xF8891400 debug_cpu_pmu1: 0xF8893400
Width	32 bits

Access Type rw
Reset Value x
Description pmevtyper0

Register PMXEVTYPER0 Details

Field Name	Bits	Type	Reset Value	Description
PMXEVTYPER0	31:0	rw	x	pmevtyper0

Register ([cortexa9_pmu](#)) PMXEVTYPER1

Name PMXEVTYPER1
Relative Address 0x00000404
Absolute Address debug_cpu_pmu0: 0xF8891404
 debug_cpu_pmu1: 0xF8893404
Width 32 bits
Access Type rw
Reset Value x
Description pmevtyper1

Register PMXEVTYPER1 Details

Field Name	Bits	Type	Reset Value	Description
PMXEVTYPER1	31:0	rw	x	pmevtyper1

Register ([cortexa9_pmu](#)) PMXEVTYPER2

Name PMXEVTYPER2
Relative Address 0x00000408
Absolute Address debug_cpu_pmu0: 0xF8891408
 debug_cpu_pmu1: 0xF8893408
Width 32 bits
Access Type rw
Reset Value x
Description pmevtyper2

Register PMXEVTYPER2 Details

Field Name	Bits	Type	Reset Value	Description
PMXEVTYPER2	31:0	rw	x	pmevtyper2

Register ([cortexa9_pmu](#)) PMXEVTYP3

Name	PMXEVTYP3
Relative Address	0x0000040C
Absolute Address	debug_cpu_pmu0: 0xF889140C debug_cpu_pmu1: 0xF889340C
Width	32 bits
Access Type	rw
Reset Value	x
Description	pmevtyper3

Register PMXEVTYP3 Details

Field Name	Bits	Type	Reset Value	Description
PMXEVTYP3	31:0	rw	x	pmevtyper3

Register ([cortexa9_pmu](#)) PMXEVTYP4

Name	PMXEVTYP4
Relative Address	0x00000410
Absolute Address	debug_cpu_pmu0: 0xF8891410 debug_cpu_pmu1: 0xF8893410
Width	32 bits
Access Type	rw
Reset Value	x
Description	pmevtyper4

Register PMXEVTYP4 Details

Field Name	Bits	Type	Reset Value	Description
PMXEVTYP4	31:0	rw	x	pmevtyper4

Register ([cortexa9_pmu](#)) PMXEVTYP5

Name	PMXEVTYP5
Relative Address	0x00000414
Absolute Address	debug_cpu_pmu0: 0xF8891414 debug_cpu_pmu1: 0xF8893414
Width	32 bits

Access Type rw
Reset Value x
Description pmevtyper5

Register PMXEVTYPER5 Details

Field Name	Bits	Type	Reset Value	Description
PMXEVTYPER5	31:0	rw	x	pmevtyper5

Register ([cortexa9_pmu](#)) PMCNTENSET

Name PMCNTENSET
Relative Address 0x00000C00
Absolute Address debug_cpu_pmu0: 0xF8891C00
 debug_cpu_pmu1: 0xF8893C00
Width 32 bits
Access Type rw
Reset Value 0x00000000
Description pmcntenset

Register PMCNTENSET Details

Field Name	Bits	Type	Reset Value	Description
PMCNTENSET	31:0	rw	0x0	pmcntenset

Register ([cortexa9_pmu](#)) PMCNTENCLR

Name PMCNTENCLR
Relative Address 0x00000C20
Absolute Address debug_cpu_pmu0: 0xF8891C20
 debug_cpu_pmu1: 0xF8893C20
Width 32 bits
Access Type rw
Reset Value 0x00000000
Description pmcntenclr

Register PMCNTENCLR Details

Field Name	Bits	Type	Reset Value	Description
PMCNTENCLR	31:0	rw	0x0	pmcntenclr

Register ([cortexa9_pmu](#)) PMINTENSET

Name	PMINTENSET
Relative Address	0x00000C40
Absolute Address	debug_cpu_pmu0: 0xF8891C40 debug_cpu_pmu1: 0xF8893C40
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	pmintenset

Register PMINTENSET Details

Field Name	Bits	Type	Reset Value	Description
PMINTENSET	31:0	rw	0x0	pmintenset

Register ([cortexa9_pmu](#)) PMINTENCLR

Name	PMINTENCLR
Relative Address	0x00000C60
Absolute Address	debug_cpu_pmu0: 0xF8891C60 debug_cpu_pmu1: 0xF8893C60
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	pmintenclr

Register PMINTENCLR Details

Field Name	Bits	Type	Reset Value	Description
PMINTENCLR	31:0	rw	0x0	pmintenclr

Register ([cortexa9_pmu](#)) PMOVSR

Name	PMOVSR
Relative Address	0x00000C80
Absolute Address	debug_cpu_pmu0: 0xF8891C80 debug_cpu_pmu1: 0xF8893C80
Width	32 bits

Access Type rw
Reset Value x
Description pmovsr

Register PMOVSR Details

Field Name	Bits	Type	Reset Value	Description
PMOVSR	31:0	rw	x	pmovsr

Register ([cortexa9_pmu](#)) PMSWINC

Name PMSWINC
Relative Address 0x00000CA0
Absolute Address debug_cpu_pmu0: 0xF8891CA0
 debug_cpu_pmu1: 0xF8893CA0
Width 32 bits
Access Type wo
Reset Value x
Description pmswinc

Register PMSWINC Details

Field Name	Bits	Type	Reset Value	Description
PMSWINC	31:0	wo	x	pmswinc

Register ([cortexa9_pmu](#)) PMCR

Name PMCR
Relative Address 0x00000E04
Absolute Address debug_cpu_pmu0: 0xF8891E04
 debug_cpu_pmu1: 0xF8893E04
Width 32 bits
Access Type rw
Reset Value 0x41093000
Description pmcr

Register PMCR Details

Field Name	Bits	Type	Reset Value	Description
PMCR	31:0	rw	0x41093000	pmcr

Register ([cortexa9_pmu](#)) PMUSERENR

Name	PMUSERENR
Relative Address	0x00000E08
Absolute Address	debug_cpu_pmu0: 0xF8891E08 debug_cpu_pmu1: 0xF8893E08
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	pmuserenr

Register PMUSERENR Details

Field Name	Bits	Type	Reset Value	Description
PMUSERENR	31:0	rw	0x0	pmuserenr This register is read-only in user mode.

B.9 CoreSight Program Trace Macrocell (ptm)

Module Name	CoreSight Program Trace Macrocell (ptm)
Base Address	0xF889C000 debug_cpu_ptm0 0xF889D000 debug_cpu_ptm1
Description	CoreSight PTM-A9 This ptm instance is for CPU 0.
Version	r1p0
Doc Version	
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
ETMCR	0x00000000	30	rw	0x00000400	Main Control Register
ETMCCR	0x00000004	32	ro	0x8D294004	Configuration Code Register
ETMTRIGGER	0x00000008	17	rw	0x00000000	Trigger Event Register
ETMSR	0x00000010	4	mixed	0x00000000	Status Register
ETMSCR	0x00000014	15	ro	0x00000000	System Configuration Register
ETMTSSCR	0x00000018	24	rw	0x00000000	TraceEnable Start/Stop Control Register
ETMTECR1	0x00000024	26	rw	0x00000000	TraceEnable Control Register 1
ETMACVR1	0x00000040	32	rw	0x00000000	Address Comparator Value Register 1
ETMACVR2	0x00000044	32	rw	0x00000000	Address Comparator Value Register 2
ETMACVR3	0x00000048	32	rw	0x00000000	Address Comparator Value Register 3
ETMACVR4	0x0000004C	32	rw	0x00000000	Address Comparator Value Register 4
ETMACVR5	0x00000050	32	rw	0x00000000	Address Comparator Value Register 5
ETMACVR6	0x00000054	32	rw	0x00000000	Address Comparator Value Register 6
ETMACVR7	0x00000058	32	rw	0x00000000	Address Comparator Value Register 7
ETMACVR8	0x0000005C	32	rw	0x00000000	Address Comparator Value Register 8

Register Name	Address	Width	Type	Reset Value	Description
ETMACTR1	0x00000080	12	mixed	0x00000001	Address Comparator Access Type Register 1
ETMACTR2	0x00000084	12	mixed	0x00000001	Address Comparator Access Type Register 2
ETMACTR3	0x00000088	12	mixed	0x00000001	Address Comparator Access Type Register 3
ETMACTR4	0x0000008C	12	mixed	0x00000001	Address Comparator Access Type Register 4
ETMACTR5	0x00000090	12	mixed	0x00000001	Address Comparator Access Type Register 5
ETMACTR6	0x00000094	12	mixed	0x00000001	Address Comparator Access Type Register 6
ETMACTR7	0x00000098	12	mixed	0x00000001	Address Comparator Access Type Register 7
ETMACTR8	0x0000009C	12	mixed	0x00000001	Address Comparator Access Type Register 8
ETMCNTRLDVR1	0x00000140	16	rw	0x00000000	Counter Reload Value Register 1
ETMCNTRLDVR2	0x00000144	16	rw	0x00000000	Counter Reload Value Register 2
ETMCNTENR1	0x00000150	18	mixed	0x00020000	Counter Enable Event Register 1
ETMCNTENR2	0x00000154	18	mixed	0x00020000	Counter Enable Event Register 2
ETMCNTRLDEVR1	0x00000160	17	rw	0x00000000	Counter Reload Event Register 1
ETMCNTRLDEVR2	0x00000164	17	rw	0x00000000	Counter Reload Event Register 2
ETMCNTRV1	0x00000170	16	rw	0x00000000	Counter Value Register 1
ETMCNTRV2	0x00000174	16	rw	0x00000000	Counter Value Register 2
ETMSQ12EVR	0x00000180	17	rw	0x00000000	Sequencer State Transition Event Register 12
ETMSQ21EVR	0x00000184	17	rw	0x00000000	Sequencer State Transition Event Register 21
ETMSQ23EVR	0x00000188	17	rw	0x00000000	Sequencer State Transition Event Register 23
ETMSQ31EVR	0x0000018C	17	rw	0x00000000	Sequencer State Transition Event Register 31
ETMSQ32EVR	0x00000190	17	rw	0x00000000	Sequencer State Transition Event Register 32
ETMSQ13EVR	0x00000194	17	rw	0x00000000	Sequencer State Transition Event Register 13
ETMSQR	0x0000019C	2	rw	0x00000000	Current Sequencer State Register
ETMEXTOUTEVR1	0x000001A0	17	rw	0x00000000	External Output Event Register 1

Register Name	Address	Width	Type	Reset Value	Description
ETMEXTOUTEVR2	0x000001A4	17	rw	0x00000000	External Output Event Register 2
ETMCIDCVR1	0x000001B0	32	rw	0x00000000	Context ID Comparator Value Register
ETMCIDCMR	0x000001BC	32	rw	0x00000000	Context ID Comparator Mask Register
ETMSYNCFR	0x000001E0	12	mixed	0x00000400	Synchronization Frequency Register
ETMIDR	0x000001E4	32	ro	0x411CF300	ID Register
ETMCCER	0x000001E8	26	ro	0x00C019A2	Configuration Code Extension Register
ETMEXTINSEL	0x000001EC	14	rw	0x00000000	Extended External Input Selection Register
ETMAUXCR	0x000001FC	4	rw	0x00000000	Auxiliary Control Register
ETMTRACEIDR	0x00000200	7	rw	0x00000000	CoreSight Trace ID Register
OSLSR	0x00000304	32	ro	0x00000000	OS Lock Status Register
ETMPDSR	0x00000314	32	ro	0x00000001	Device Powerdown Status Register
ITMISCOUT	0x00000EDC	10	wo	0x00000000	Miscellaneous Outputs Register
ITMISCIN	0x00000EE0	7	ro	x	Miscellaneous Inputs Register
ITTRIGGER	0x00000EE8	1	wo	0x00000000	Trigger Register
ITATBDATA0	0x00000EEC	5	wo	0x00000000	ATB Data 0 Register
ITATBCTR2	0x00000EF0	2	ro	x	ATB Control 2 Register
ITATBID	0x00000EF4	7	wo	0x00000000	ATB Identification Register
ITATBCTR0	0x00000EF8	10	wo	0x00000000	ATB Control 0 Register
ETMITCTRL	0x00000F00	1	rw	0x00000000	Integration Mode Control Register
CTSR	0x00000FA0	8	rw	0x000000FF	Claim Tag Set Register
CTCR	0x00000FA4	8	rw	0x00000000	Claim Tag Clear Register
LAR	0x00000FB0	32	wo	0x00000000	Lock Access Register
LSR	0x00000FB4	3	ro	0x00000003	Lock Status Register
ASR	0x00000FB8	8	ro	x	Authentication Status Register
DEVID	0x00000FC8	32	ro	0x00000000	Device ID
DTIR	0x00000FCC	8	ro	0x00000013	Device Type Identifier Register
PERIPHID4	0x00000FD0	8	ro	0x00000004	Peripheral ID4
PERIPHID5	0x00000FD4	8	ro	0x00000000	Peripheral ID5
PERIPHID6	0x00000FD8	8	ro	0x00000000	Peripheral ID6

Register Name	Address	Width	Type	Reset Value	Description
PERIPHID7	0x00000FDC	8	ro	0x00000000	Peripheral ID7
PERIPHID0	0x00000FE0	8	ro	0x00000050	Peripheral ID0
PERIPHID1	0x00000FE4	8	ro	0x000000B9	Peripheral ID1
PERIPHID2	0x00000FE8	8	ro	0x0000001B	Peripheral ID2
PERIPHID3	0x00000FEC	8	ro	0x00000000	Peripheral ID3
COMPID0	0x00000FF0	8	ro	0x0000000D	Component ID0
COMPID1	0x00000FF4	8	ro	0x00000090	Component ID1
COMPID2	0x00000FF8	8	ro	0x00000005	Component ID2
COMPID3	0x00000FFC	8	ro	0x000000B1	Component ID3

Register ([ptm](#)) ETMCR

Name	ETMCR
Relative Address	0x00000000
Absolute Address	debug_cpu_ptm0: 0xF889C000 debug_cpu_ptm1: 0xF889D000
Width	30 bits
Access Type	rw
Reset Value	0x00000400
Description	Main Control Register

Register ETMCR Details

Field Name	Bits	Type	Reset Value	Description
ReturnStackEn	29	rw	0x0	Return stack enable
TimestampEn	28	rw	0x0	Timestamp enable
ProcSelect	27:25	rw	0x0	Select for external multiplexor if PTM is shared between multiple processors.
reserved	24:16	rw	0x0	Reserved
ContextIDSize	15:14	rw	0x0	Context ID Size Enumerated Value List: NONE=0. 8BIT=1. 16BIT=2. 32BIT=3.
reserved	13	rw	0x0	Reserved
CycleAccurate	12	rw	0x0	Enables cycle counting

Field Name	Bits	Type	Reset Value	Description
reserved	11	rw	0x0	Reserved
ProgBit	10	rw	0x1	This bit must be set to b1 when the PTM is being programmed.
DebugReqCtrl	9	rw	0x0	Debug Request Control When set to b1 and the trigger event occurs, the PTMDBGRQ output is asserted until PTMDBGACK is observed. This enables a debugger to force the processor into Debug state.
BranchOutput	8	rw	0x0	When this bit is set to b1, addresses are output for all executed branches, both direct and indirect.
reserved	7:1	rw	0x0	Reserved
PowerDown	0	rw	0x0	This bit enables external control of the PTM. This bit must be cleared by the trace software tools at the beginning of a debug session. When this bit is set to b0, both the PTM and the trace interface in the processor are enabled. To avoid corruption of trace data, this bit must not be set before the Programming Status bit in the PTM Status Register has been read as 1.

Register ([ptm](#)) ETMCCR

Name	ETMCCR
Relative Address	0x00000004
Absolute Address	debug_cpu_ptm0: 0xF889C004 debug_cpu_ptm1: 0xF889D004
Width	32 bits
Access Type	ro
Reset Value	0x8D294004
Description	Configuration Code Register

Register ETMCCR Details

Field Name	Bits	Type	Reset Value	Description
IDRegPresent	31	ro	0x1	Indicates that the ID Register is present.
reserved	30:28	ro	0x0	Reserved
SoftwareAccess	27	ro	0x1	Indicates that software access is supported.
TraceSSB	26	ro	0x1	Indicates that the trace start/stop block is present.
NumCntxtIDComp	25:24	ro	0x1	Specifies the number of Context ID comparators, one.

Field Name	Bits	Type	Reset Value	Description
FIFOFULLLogic	23	ro	0x0	Indicates that it is not possible to stall the processor to prevent FIFO overflow.
NumExtOut	22:20	ro	0x2	Specifies the number of external outputs, two.
NumExtIn	19:17	ro	0x4	Specifies the number of external inputs, four.
Sequencer	16	ro	0x1	Indicates that the sequencer is present.
NumCounters	15:13	ro	0x2	Specifies the number of counters, two.
reserved	12:4	ro	0x0	Reserved
NumAddrComp	3:0	ro	0x4	Specifies the number of address comparator pairs, four.

Register ([ptm](#)) ETMTRIGGER

Name	ETMTRIGGER
Relative Address	0x00000008
Absolute Address	debug_cpu_ptm0: 0xF889C008 debug_cpu_ptm1: 0xF889D008
Width	17 bits
Access Type	rw
Reset Value	0x00000000
Description	Trigger Event Register

Register ETMTRIGGER Details

Field Name	Bits	Type	Reset Value	Description
TrigEvent	16:0	rw	0x0	Trigger event. Subdivided as: Function, bits [16:14] Specifies the function that combines the two resources that define the event. Resource B, bits [13:7] and Resource A, bits [6:0] Specify the two resources that are combined by the logical operation specified by the Function field.

Register ([ptm](#)) ETMSR

Name	ETMSR
Relative Address	0x00000010
Absolute Address	debug_cpu_ptm0: 0xF889C010 debug_cpu_ptm1: 0xF889D010

Width	4 bits
Access Type	mixed
Reset Value	0x00000000
Description	Status Register

Register ETMSR Details

Field Name	Bits	Type	Reset Value	Description
TrigFlag	3	rw	0x0	Trigger bit. Set when the trigger occurs and prevents the trigger from being output until the PTM is next programmed.
TSSRStat	2	rw	0x0	Holds the current status of the trace start/stop resource. If set to 1, indicates that a trace start address has been matched, without a corresponding trace stop address match.
ProgBit	1	ro	0x0	Effective state of the Programming bit. You must wait for this bit to go to b1 before starting to program the PTM.
Overflow	0	ro	0x0	If set to 1, there is an overflow that has not yet been traced.

Register ([ptm](#)) ETMSCR

Name	ETMSCR
Relative Address	0x00000014
Absolute Address	debug_cpu_ptm0: 0xF889C014 debug_cpu_ptm1: 0xF889D014
Width	15 bits
Access Type	ro
Reset Value	0x00000000
Description	System Configuration Register

Register ETMSCR Details

Field Name	Bits	Type	Reset Value	Description
NumProcs	14:12	ro	0x0	Number of supported processors minus 1.
reserved	11:9	ro	0x0	Reserved
FIFOFULL	8	ro	0x0	FIFOFULL not supported
reserved	7:0	ro	0x0	Reserved

Register ([ptm](#)) ETMTSSCR

Name	ETMTSSCR
Relative Address	0x00000018
Absolute Address	debug_cpu_ptm0: 0xF889C018 debug_cpu_ptm1: 0xF889D018
Width	24 bits
Access Type	rw
Reset Value	0x00000000
Description	TraceEnable Start/Stop Control Register

Register ETMTSSCR Details

Field Name	Bits	Type	Reset Value	Description
StopAddrSel	23:16	rw	0x0	When a bit is set to 1, it selects a single address comparator (8-1) as a stop address for the TraceEnable Start/Stop block. For example, if you set bit [16] to 1 it selects single address comparator 1 as a stop address.
reserved	15:8	rw	0x0	Reserved
StartAddrSel	7:0	rw	0x0	When a bit is set to 1, it selects a single address comparator (8-1) as a start address for the TraceEnable Start/Stop block. For example, if you set bit [0] to 1 it selects single address comparator 1 as a start address.

Register ([ptm](#)) ETMTECR1

Name	ETMTECR1
Relative Address	0x00000024
Absolute Address	debug_cpu_ptm0: 0xF889C024 debug_cpu_ptm1: 0xF889D024
Width	26 bits
Access Type	rw
Reset Value	0x00000000
Description	TraceEnable Control Register 1

Register ETMTECR1 Details

Field Name	Bits	Type	Reset Value	Description
TraceSSEn	25	rw	0x0	Trace start/stop control enable. The possible values of this bit are: 0 Tracing is unaffected by the trace start/stop logic. 1 Tracing is controlled by the trace on and off addresses configured for the trace start/stop logic. The trace start/stop resource is not affected by the value of this bit.
ExcIncFlag	24	rw	0x0	Exclude/include flag. The possible values of this bit are: 0 Include. The specified address range comparators indicate the regions where tracing can occur. No tracing occurs outside this region. 1 Exclude. The specified address range comparators indicate regions to be excluded from the trace. When outside an exclude region, tracing can occur.
reserved	23:4	rw	0x0	Reserved
AddrCompSel	3:0	rw	0x0	When a bit is set to 1, it selects an address range comparator, 4-1, for include/exclude control. For example, bit [0] set to 1 selects address range comparator 1.

Register ([ptm](#)) ETMACVR1

Name	ETMACVR1
Relative Address	0x00000040
Absolute Address	debug_cpu_ptm0: 0xF889C040 debug_cpu_ptm1: 0xF889D040
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Address Comparator Value Register 1

Register ETMACVR1 Details

Field Name	Bits	Type	Reset Value	Description
Address	31:0	rw	0x0	Address for comparison

Register ([ptm](#)) ETMACVR2

Name	ETMACVR2
Relative Address	0x00000044
Absolute Address	debug_cpu_ptm0: 0xF889C044 debug_cpu_ptm1: 0xF889D044
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Address Comparator Value Register 2

Register ETMACVR2 Details

Field Name	Bits	Type	Reset Value	Description
Address	31:0	rw	0x0	Address for comparison

Register ([ptm](#)) ETMACVR3

Name	ETMACVR3
Relative Address	0x00000048
Absolute Address	debug_cpu_ptm0: 0xF889C048 debug_cpu_ptm1: 0xF889D048
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Address Comparator Value Register 3

Register ETMACVR3 Details

Field Name	Bits	Type	Reset Value	Description
Address	31:0	rw	0x0	Address for comparison

Register ([ptm](#)) ETMACVR4

Name	ETMACVR4
Relative Address	0x0000004C
Absolute Address	debug_cpu_ptm0: 0xF889C04C debug_cpu_ptm1: 0xF889D04C
Width	32 bits

Access Type rw
Reset Value 0x00000000
Description Address Comparator Value Register 4

Register ETMACVR4 Details

Field Name	Bits	Type	Reset Value	Description
Address	31:0	rw	0x0	Address for comparison

Register ([ptm](#)) ETMACVR5

Name ETMACVR5
Relative Address 0x00000050
Absolute Address debug_cpu_ptm0: 0xF889C050
debug_cpu_ptm1: 0xF889D050
Width 32 bits
Access Type rw
Reset Value 0x00000000
Description Address Comparator Value Register 5

Register ETMACVR5 Details

Field Name	Bits	Type	Reset Value	Description
Address	31:0	rw	0x0	Address for comparison

Register ([ptm](#)) ETMACVR6

Name ETMACVR6
Relative Address 0x00000054
Absolute Address debug_cpu_ptm0: 0xF889C054
debug_cpu_ptm1: 0xF889D054
Width 32 bits
Access Type rw
Reset Value 0x00000000
Description Address Comparator Value Register 6

Register ETMACVR6 Details

Field Name	Bits	Type	Reset Value	Description
Address	31:0	rw	0x0	Address for comparison

Register ([ptm](#)) ETMACVR7

Name	ETMACVR7
Relative Address	0x00000058
Absolute Address	debug_cpu_ptm0: 0xF889C058 debug_cpu_ptm1: 0xF889D058
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Address Comparator Value Register 7

Register ETMACVR7 Details

Field Name	Bits	Type	Reset Value	Description
Address	31:0	rw	0x0	Address for comparison

Register ([ptm](#)) ETMACVR8

Name	ETMACVR8
Relative Address	0x0000005C
Absolute Address	debug_cpu_ptm0: 0xF889C05C debug_cpu_ptm1: 0xF889D05C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Address Comparator Value Register 8

Register ETMACVR8 Details

Field Name	Bits	Type	Reset Value	Description
Address	31:0	rw	0x0	Address for comparison

Register ([ptm](#)) ETMACTR1

Name	ETMACTR1
Relative Address	0x00000080
Absolute Address	debug_cpu_ptm0: 0xF889C080 debug_cpu_ptm1: 0xF889D080
Width	12 bits

Access Type mixed

Reset Value 0x00000001

Description Address Comparator Access Type Register 1

Register ETMACTR1 Details

Field Name	Bits	Type	Reset Value	Description
SecLevelCtrl	11:10	rw	0x0	Security level control Enumerated Value List: IGNORE=0. NONSEC=1. SECURE=2.
ContextIDCompCtrl	9:8	rw	0x0	Context ID comparator control. Enumerated Value List: IGNORE=0. MATCH1=1. MATCH2=2. MATCH3=3.
reserved	7:3	rw	0x0	Reserved
AccessType	2:0	ro	0x1	Access type. Returns the value: Instruction execute.

Register ([ptm](#)) ETMACTR2

Name ETMACTR2

Relative Address 0x00000084

Absolute Address debug_cpu_ptm0: 0xF889C084
 debug_cpu_ptm1: 0xF889D084

Width 12 bits

Access Type mixed

Reset Value 0x00000001

Description Address Comparator Access Type Register 2

Register ETMACTR2 Details

Field Name	Bits	Type	Reset Value	Description
SecLevelCtrl	11:10	rw	0x0	Security level control Enumerated Value List: IGNORE=0. NONSEC=1. SECURE=2.
ContextIDCompCtrl	9:8	rw	0x0	Context ID comparator control. Enumerated Value List: IGNORE=0. MATCH1=1. MATCH2=2. MATCH3=3.
reserved	7:3	rw	0x0	Reserved
AccessType	2:0	ro	0x1	Access type. Returns the value: Instruction execute.

Register ([ptm](#)) ETMACTR3

Name	ETMACTR3
Relative Address	0x00000088
Absolute Address	debug_cpu_ptm0: 0xF889C088 debug_cpu_ptm1: 0xF889D088
Width	12 bits
Access Type	mixed
Reset Value	0x00000001
Description	Address Comparator Access Type Register 3

Register ETMACTR3 Details

Field Name	Bits	Type	Reset Value	Description
SecLevelCtrl	11:10	rw	0x0	Security level control Enumerated Value List: IGNORE=0. NONSEC=1. SECURE=2.
ContextIDCompCtrl	9:8	rw	0x0	Context ID comparator control. Enumerated Value List: IGNORE=0. MATCH1=1. MATCH2=2. MATCH3=3.
reserved	7:3	rw	0x0	Reserved
AccessType	2:0	ro	0x1	Access type. Returns the value: Instruction execute.

Register ([ptm](#)) ETMACTR4

Name	ETMACTR4
Relative Address	0x0000008C
Absolute Address	debug_cpu_ptm0: 0xF889C08C debug_cpu_ptm1: 0xF889D08C
Width	12 bits
Access Type	mixed
Reset Value	0x00000001
Description	Address Comparator Access Type Register 4

Register ETMACTR4 Details

Field Name	Bits	Type	Reset Value	Description
SecLevelCtrl	11:10	rw	0x0	Security level control Enumerated Value List: IGNORE=0. NONSEC=1. SECURE=2.
ContextIDCompCtrl	9:8	rw	0x0	Context ID comparator control. Enumerated Value List: IGNORE=0. MATCH1=1. MATCH2=2. MATCH3=3.
reserved	7:3	rw	0x0	Reserved
AccessType	2:0	ro	0x1	Access type. Returns the value: Instruction execute.

Register ([ptm](#)) ETMACTR5

Name	ETMACTR5
Relative Address	0x00000090
Absolute Address	debug_cpu_ptm0: 0xF889C090 debug_cpu_ptm1: 0xF889D090
Width	12 bits
Access Type	mixed
Reset Value	0x00000001
Description	Address Comparator Access Type Register 5

Register ETMACTR5 Details

Field Name	Bits	Type	Reset Value	Description
SecLevelCtrl	11:10	rw	0x0	Security level control Enumerated Value List: IGNORE=0. NONSEC=1. SECURE=2.
ContextIDCompCtrl	9:8	rw	0x0	Context ID comparator control. Enumerated Value List: IGNORE=0. MATCH1=1. MATCH2=2. MATCH3=3.
reserved	7:3	rw	0x0	Reserved
AccessType	2:0	ro	0x1	Access type. Returns the value: Instruction execute.

Register ([ptm](#)) ETMACTR6

Name	ETMACTR6
Relative Address	0x00000094
Absolute Address	debug_cpu_ptm0: 0xF889C094 debug_cpu_ptm1: 0xF889D094
Width	12 bits
Access Type	mixed
Reset Value	0x00000001
Description	Address Comparator Access Type Register 6

Register ETMACTR6 Details

Field Name	Bits	Type	Reset Value	Description
SecLevelCtrl	11:10	rw	0x0	Security level control Enumerated Value List: IGNORE=0. NONSEC=1. SECURE=2.
ContextIDCompCtrl	9:8	rw	0x0	Context ID comparator control. Enumerated Value List: IGNORE=0. MATCH1=1. MATCH2=2. MATCH3=3.
reserved	7:3	rw	0x0	Reserved
AccessType	2:0	ro	0x1	Access type. Returns the value: Instruction execute.

Register ([ptm](#)) ETMACTR7

Name	ETMACTR7
Relative Address	0x00000098
Absolute Address	debug_cpu_ptm0: 0xF889C098 debug_cpu_ptm1: 0xF889D098
Width	12 bits
Access Type	mixed
Reset Value	0x00000001
Description	Address Comparator Access Type Register 7

Register ETMACTR7 Details

Field Name	Bits	Type	Reset Value	Description
SecLevelCtrl	11:10	rw	0x0	Security level control Enumerated Value List: IGNORE=0. NONSEC=1. SECURE=2.
ContextIDCompCtrl	9:8	rw	0x0	Context ID comparator control. Enumerated Value List: IGNORE=0. MATCH1=1. MATCH2=2. MATCH3=3.
reserved	7:3	rw	0x0	Reserved
AccessType	2:0	ro	0x1	Access type. Returns the value: Instruction execute.

Register ([ptm](#)) ETMACTR8

Name	ETMACTR8
Relative Address	0x0000009C
Absolute Address	debug_cpu_ptm0: 0xF889C09C debug_cpu_ptm1: 0xF889D09C
Width	12 bits
Access Type	mixed
Reset Value	0x00000001
Description	Address Comparator Access Type Register 8

Register ETMACTR8 Details

Field Name	Bits	Type	Reset Value	Description
SecLevelCtrl	11:10	rw	0x0	Security level control Enumerated Value List: IGNORE=0. NONSEC=1. SECURE=2.
ContextIDCompCtrl	9:8	rw	0x0	Context ID comparator control. Enumerated Value List: IGNORE=0. MATCH1=1. MATCH2=2. MATCH3=3.
reserved	7:3	rw	0x0	Reserved
AccessType	2:0	ro	0x1	Access type. Returns the value: Instruction execute.

Register ([ptm](#)) ETMCNTRLDVR1

Name	ETMCNTRLDVR1
Relative Address	0x00000140
Absolute Address	debug_cpu_ptm0: 0xF889C140 debug_cpu_ptm1: 0xF889D140
Width	16 bits
Access Type	rw
Reset Value	0x00000000
Description	Counter Reload Value Register 1

Register ETMCNTRLDVR1 Details

Field Name	Bits	Type	Reset Value	Description
InitValue	15:0	rw	0x0	Counter initial value

Register ([ptm](#)) ETMCNTRLDVR2

Name	ETMCNTRLDVR2
Relative Address	0x00000144
Absolute Address	debug_cpu_ptm0: 0xF889C144 debug_cpu_ptm1: 0xF889D144

Width	16 bits
Access Type	rw
Reset Value	0x00000000
Description	Counter Reload Value Register 2

Register ETMCNTRLDVR2 Details

Field Name	Bits	Type	Reset Value	Description
InitValue	15:0	rw	0x0	Counter initial value

Register ([ptm](#)) ETMCNTENR1

Name	ETMCNTENR1
Relative Address	0x00000150
Absolute Address	debug_cpu_ptm0: 0xF889C150 debug_cpu_ptm1: 0xF889D150
Width	18 bits
Access Type	mixed
Reset Value	0x00020000
Description	Counter Enable Event Register 1

Register ETMCNTENR1 Details

Field Name	Bits	Type	Reset Value	Description
Reserved_1	17	ro	0x1	Reserved, RAO/WI
ExtOutEvent	16:0	rw	0x0	Count enable event. Subdivided as: Function, bits [16:14] Specifies the function that combines the two resources that define the event. Resource B, bits [13:7] and Resource A, bits [6:0] Specify the two resources that are combined by the logical operation specified by the Function field.

Register ([ptm](#)) ETMCNTENR2

Name	ETMCNTENR2
Relative Address	0x00000154
Absolute Address	debug_cpu_ptm0: 0xF889C154 debug_cpu_ptm1: 0xF889D154

Width	18 bits
Access Type	mixed
Reset Value	0x00020000
Description	Counter Enable Event Register 2

Register ETMCNTENR2 Details

Field Name	Bits	Type	Reset Value	Description
Reserved_1	17	ro	0x1	Reserved, RAO/WI
ExtOutEvent	16:0	rw	0x0	Count enable event. Subdivided as: Function, bits [16:14] Specifies the function that combines the two resources that define the event. Resource B, bits [13:7] and Resource A, bits [6:0] Specify the two resources that are combined by the logical operation specified by the Function field.

Register ([ptm](#)) ETMCNTRLDEVR1

Name	ETMCNTRLDEVR1
Relative Address	0x00000160
Absolute Address	debug_cpu_ptm0: 0xF889C160 debug_cpu_ptm1: 0xF889D160
Width	17 bits
Access Type	rw
Reset Value	0x00000000
Description	Counter Reload Event Register 1

Register ETMCNTRLDEVR1 Details

Field Name	Bits	Type	Reset Value	Description
CntReloadEvent	16:0	rw	0x0	Count reload event. Subdivided as: Function, bits [16:14] Specifies the function that combines the two resources that define the event. Resource B, bits [13:7] and Resource A, bits [6:0] Specify the two resources that are combined by the logical operation specified by the Function field.

Register ([ptm](#)) ETMCNTRLDEVR2

Name	ETMCNTRLDEVR2
Relative Address	0x00000164
Absolute Address	debug_cpu_ptm0: 0xF889C164 debug_cpu_ptm1: 0xF889D164
Width	17 bits
Access Type	rw
Reset Value	0x00000000
Description	Counter Reload Event Register 2

Register ETMCNTRLDEVR2 Details

Field Name	Bits	Type	Reset Value	Description
CntReloadEvent	16:0	rw	0x0	Count reload event. Subdivided as: Function, bits [16:14] Specifies the function that combines the two resources that define the event. Resource B, bits [13:7] and Resource A, bits [6:0] Specify the two resources that are combined by the logical operation specified by the Function field.

Register ([ptm](#)) ETMCNTVR1

Name	ETMCNTVR1
Relative Address	0x00000170
Absolute Address	debug_cpu_ptm0: 0xF889C170 debug_cpu_ptm1: 0xF889D170
Width	16 bits
Access Type	rw
Reset Value	0x00000000
Description	Counter Value Register 1

Register ETMCNTVR1 Details

Field Name	Bits	Type	Reset Value	Description
CurrCount	15:0	rw	0x0	Current counter value.

Register ([ptm](#)) ETMCNTVR2

Name	ETMCNTVR2
Relative Address	0x00000174
Absolute Address	debug_cpu_ptm0: 0xF889C174 debug_cpu_ptm1: 0xF889D174
Width	16 bits
Access Type	rw
Reset Value	0x00000000
Description	Counter Value Register 2

Register ETMCNTVR2 Details

Field Name	Bits	Type	Reset Value	Description
CurrCount	15:0	rw	0x0	Current counter value.

Register ([ptm](#)) ETMSQ12EVR

Name	ETMSQ12EVR
Relative Address	0x00000180
Absolute Address	debug_cpu_ptm0: 0xF889C180 debug_cpu_ptm1: 0xF889D180
Width	17 bits
Access Type	rw
Reset Value	0x00000000
Description	Sequencer State Transition Event Register 12

Register ETMSQ12EVR Details

Field Name	Bits	Type	Reset Value	Description
TransEvent	16:0	rw	0x0	<p>A Sequencer State Transition Event Register, ETMSQmnEVR, defines the event that causes the sequencer state transition from state m to state n. The format is subdivided as:</p> <p>Function, bits [16:14] Specifies the function that combines the two resources that define the event.</p> <p>Resource B, bits [13:7] and Resource A, bits [6:0] Specify the two resources that are combined by the logical operation specified by the Function field.</p>

Register ([ptm](#)) ETMSQ21EVR

Name	ETMSQ21EVR
Relative Address	0x00000184
Absolute Address	debug_cpu_ptm0: 0xF889C184 debug_cpu_ptm1: 0xF889D184
Width	17 bits
Access Type	rw
Reset Value	0x00000000
Description	Sequencer State Transition Event Register 21

Register ETMSQ21EVR Details

Field Name	Bits	Type	Reset Value	Description
TransEvent	16:0	rw	0x0	<p>A Sequencer State Transition Event Register, ETMSQmnEVR, defines the event that causes the sequencer state transition from state m to state n. The format is subdivided as:</p> <p>Function, bits [16:14] Specifies the function that combines the two resources that define the event.</p> <p>Resource B, bits [13:7] and Resource A, bits [6:0] Specify the two resources that are combined by the logical operation specified by the Function field.</p>

Register ([ptm](#)) ETMSQ23EVR

Name	ETMSQ23EVR
Relative Address	0x00000188
Absolute Address	debug_cpu_ptm0: 0xF889C188 debug_cpu_ptm1: 0xF889D188
Width	17 bits
Access Type	rw
Reset Value	0x00000000
Description	Sequencer State Transition Event Register 23

Register ETMSQ23EVR Details

Field Name	Bits	Type	Reset Value	Description
TransEvent	16:0	rw	0x0	<p>A Sequencer State Transition Event Register, ETMSQmnEVR, defines the event that causes the sequencer state transition from state m to state n. The format is subdivided as:</p> <p>Function, bits [16:14]</p> <p>Specifies the function that combines the two resources that define the event.</p> <p>Resource B, bits [13:7] and Resource A, bits [6:0]</p> <p>Specify the two resources that are combined by the logical operation specified by the Function field.</p>

Register ([ptm](#)) ETMSQ31EVR

Name	ETMSQ31EVR
Relative Address	0x0000018C
Absolute Address	debug_cpu_ptm0: 0xF889C18C debug_cpu_ptm1: 0xF889D18C
Width	17 bits
Access Type	rw
Reset Value	0x00000000
Description	Sequencer State Transition Event Register 31

Register ETMSQ31EVR Details

Field Name	Bits	Type	Reset Value	Description
TransEvent	16:0	rw	0x0	<p>A Sequencer State Transition Event Register, ETMSQmnEVR, defines the event that causes the sequencer state transition from state m to state n. The format is subdivided as:</p> <p>Function, bits [16:14]</p> <p>Specifies the function that combines the two resources that define the event.</p> <p>Resource B, bits [13:7] and Resource A, bits [6:0]</p> <p>Specify the two resources that are combined by the logical operation specified by the Function field.</p>

Register ([ptm](#)) ETMSQ32EVR

Name	ETMSQ32EVR
------	------------

Relative Address	0x00000190
Absolute Address	debug_cpu_ptm0: 0xF889C190 debug_cpu_ptm1: 0xF889D190
Width	17 bits
Access Type	rw
Reset Value	0x00000000
Description	Sequencer State Transition Event Register 32

Register ETMSQ32EVR Details

Field Name	Bits	Type	Reset Value	Description
TransEvent	16:0	rw	0x0	A Sequencer State Transition Event Register, ETMSQmnEVR, defines the event that causes the sequencer state transition from state m to state n. The format is subdivided as: Function, bits [16:14] Specifies the function that combines the two resources that define the event. Resource B, bits [13:7] and Resource A, bits [6:0] Specify the two resources that are combined by the logical operation specified by the Function field.

Register ([ptm](#)) ETMSQ13EVR

Name	ETMSQ13EVR
Relative Address	0x00000194
Absolute Address	debug_cpu_ptm0: 0xF889C194 debug_cpu_ptm1: 0xF889D194
Width	17 bits
Access Type	rw
Reset Value	0x00000000
Description	Sequencer State Transition Event Register 13

Register ETMSQ13EVR Details

Field Name	Bits	Type	Reset Value	Description
TransEvent	16:0	rw	0x0	<p>A Sequencer State Transition Event Register, ETMSQmnEVR, defines the event that causes the sequencer state transition from state m to state n. The format is subdivided as:</p> <p>Function, bits [16:14] Specifies the function that combines the two resources that define the event.</p> <p>Resource B, bits [13:7] and Resource A, bits [6:0] Specify the two resources that are combined by the logical operation specified by the Function field.</p>

Register ([ptm](#)) ETMSQR

Name	ETMSQR
Relative Address	0x0000019C
Absolute Address	debug_cpu_ptm0: 0xF889C19C debug_cpu_ptm1: 0xF889D19C
Width	2 bits
Access Type	rw
Reset Value	0x00000000
Description	Current Sequencer State Register

Register ETMSQR Details

Field Name	Bits	Type	Reset Value	Description
CurrentSeqState	1:0	rw	0x0	Indicates the current sequencer state

Register ([ptm](#)) ETMEXTOUTEVR1

Name	ETMEXTOUTEVR1
Relative Address	0x000001A0
Absolute Address	debug_cpu_ptm0: 0xF889C1A0 debug_cpu_ptm1: 0xF889D1A0
Width	17 bits
Access Type	rw
Reset Value	0x00000000
Description	External Output Event Register 1

Register ETMEXTOUTEVR1 Details

Field Name	Bits	Type	Reset Value	Description
ExtOutputEvent	16:0	rw	0x0	External output event. Subdivided as: Function, bits [16:14] Specifies the function that combines the two resources that define the event. Resource B, bits [13:7] and Resource A, bits [6:0] Specify the two resources that are combined by the logical operation specified by the Function field.

Register ([ptm](#)) ETMEXTOUTEVR2

Name	ETMEXTOUTEVR2
Relative Address	0x000001A4
Absolute Address	debug_cpu_ptm0: 0xF889C1A4 debug_cpu_ptm1: 0xF889D1A4
Width	17 bits
Access Type	rw
Reset Value	0x00000000
Description	External Output Event Register 2

Register ETMEXTOUTEVR2 Details

Field Name	Bits	Type	Reset Value	Description
ExtOutputEvent	16:0	rw	0x0	External output event. Subdivided as: Function, bits [16:14] Specifies the function that combines the two resources that define the event. Resource B, bits [13:7] and Resource A, bits [6:0] Specify the two resources that are combined by the logical operation specified by the Function field.

Register ([ptm](#)) ETMCIDCVR1

Name	ETMCIDCVR1
Relative Address	0x000001B0
Absolute Address	debug_cpu_ptm0: 0xF889C1B0 debug_cpu_ptm1: 0xF889D1B0
Width	32 bits

Access Type rw

Reset Value 0x00000000

Description Context ID Comparator Value Register

Register ETMCIDCVR1 Details

Field Name	Bits	Type	Reset Value	Description
ContextID	31:0	rw	0x0	Holds a 32-bit Context ID value

Register ([ptm](#)) ETMCIDCMR

Name ETMCIDCMR

Relative Address 0x000001BC

Absolute Address debug_cpu_ptm0: 0xF889C1BC
debug_cpu_ptm1: 0xF889D1BC

Width 32 bits

Access Type rw

Reset Value 0x00000000

Description Context ID Comparator Mask Register

Register ETMCIDCMR Details

Field Name	Bits	Type	Reset Value	Description
ContextMask	31:0	rw	0x0	Holds a 32-bit Context ID mask

Register ([ptm](#)) ETMSYNCFR

Name ETMSYNCFR

Relative Address 0x000001E0

Absolute Address debug_cpu_ptm0: 0xF889C1E0
debug_cpu_ptm1: 0xF889D1E0

Width 12 bits

Access Type mixed

Reset Value 0x00000400

Description Synchronization Frequency Register

Register ETMSYNCFR Details

Field Name	Bits	Type	Reset Value	Description
SyncFreq	11:2	rw	0x100	Synchronization frequency
reserved	1:0	ro	0x0	Reserved

Register ([ptm](#)) ETMIDR

Name	ETMIDR
Relative Address	0x000001E4
Absolute Address	debug_cpu_ptm0: 0xF889C1E4 debug_cpu_ptm1: 0xF889D1E4
Width	32 bits
Access Type	ro
Reset Value	0x411CF300
Description	ID Register

Register ETMIDR Details

Field Name	Bits	Type	Reset Value	Description
ImplCode	31:24	ro	0x41	Implementor code. The field reads 0x41, ASCII code for A, indicating ARM Limited.
reserved	23:21	ro	0x0	Reserved
reserved	20	ro	0x1	Reserved, RAO
SecExtSupp	19	ro	0x1	Support for security extensions.
Thumb32Supp	18	ro	0x1	Support for 32-bit Thumb instructions.
reserved	17:16	ro	0x0	Reserved
Reserved_F	15:12	ro	0xF	Reserved, 0b1111
MajorVer	11:8	ro	0x3	Major architecture version number, 0b0011
MinorVer	7:4	ro	0x0	Minor architecture version number, 0b0000
ImplRev	3:0	ro	0x0	Implementation revision.

Register ([ptm](#)) ETMCCER

Name	ETMCCER
Relative Address	0x000001E8
Absolute Address	debug_cpu_ptm0: 0xF889C1E8 debug_cpu_ptm1: 0xF889D1E8
Width	26 bits

Access Type ro

Reset Value 0x00C019A2

Description Configuration Code Extension Register

Register ETMCCER Details

Field Name	Bits	Type	Reset Value	Description
BarrTS	25	ro	0x0	Timestamps are not generated for DMB/DSB
BarrWP	24	ro	0x0	DMB/DSB instructions are not treated as waypoints.
RetStack	23	ro	0x1	Return stack implemented.
Timestamp	22	ro	0x1	Timestamping implemented.
reserved	21:16	ro	0x0	Reserved
InstrumRes	15:13	ro	0x0	Specifies the number of instrumentation resources.
Reserved_1	12	ro	0x1	Reserved, RAO
RegReads	11	ro	0x1	Indicates that all registers, except some Integration Test Registers, are readable.
ExtInSize	10:3	ro	0x34	Specifies the size of the extended external input bus, 52.
ExtInSel	2:0	ro	0x2	Specifies the number of extended external input selectors, 2.

Register ([ptm](#)) ETMEXTINSELR

Name ETMEXTINSELR

Relative Address 0x000001EC

Absolute Address debug_cpu_ptm0: 0xF889C1EC
debug_cpu_ptm1: 0xF889D1EC

Width 14 bits

Access Type rw

Reset Value 0x00000000

Description Extended External Input Selection Register

Register ETMEXTINSELR Details

Field Name	Bits	Type	Reset Value	Description
ExtInSel2	13:8	rw	0x0	Second extended external input selector
reserved	7:6	rw	0x0	Reserved
ExtInSel1	5:0	rw	0x0	First extended external input selector

Register ([ptm](#)) ETMAUXCR

Name	ETMAUXCR
Relative Address	0x000001FC
Absolute Address	debug_cpu_ptm0: 0xF889C1FC debug_cpu_ptm1: 0xF889D1FC
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	Auxiliary Control Register

Register ETMAUXCR Details

Field Name	Bits	Type	Reset Value	Description
ForceSyncInsert	3	rw	0x0	Force insertion of synchronization packets, regardless of current trace activity. Possible values for this bit are: b0 = Synchronization packets delayed when trace activity is high. This is the reset value. b1 = Synchronization packets inserted regardless of trace activity. This bit might be set if synchronization packets occur too far apart. Setting this bit might cause the trace FIFO to overflow more frequently when trace activity is high.
DisableWPUpdate	2	rw	0x0	Specifies whether the PTM issues waypoint update packets if there are more than 4096 bytes between waypoints. Possible values for this bit are: b0 = PTM always issues update packets if there are more than 4096 bytes between waypoints. This is the reset value. b1 = PTM does not issue waypoint update packets unless required to do so as the result of an exception or debug entry.

Field Name	Bits	Type	Reset Value	Description
DisableTSOnBarr	1	rw	0x0	Specifies whether the PTM issues a timestamp on a barrier instruction. Possible values for this bit are: b0 = PTM issues timestamps on barrier instructions. This is the reset value. b1 = PTM does not issue timestamps on barriers
DisableForcedOF	0	rw	0x0	Specifies whether the PTM enters overflow state when synchronization is requested, and the previous synchronization sequence has not yet completed. This does not affect entry to overflow state when the FIFO becomes full. Possible values for this bit are: b0 = Forced overflow enabled. This is the reset value. b1 = Forced overflow disabled.

Register ([ptm](#)) ETMTRACEIDR

Name	ETMTRACEIDR
Relative Address	0x00000200
Absolute Address	debug_cpu_ptm0: 0xF889C200 debug_cpu_ptm1: 0xF889D200
Width	7 bits
Access Type	rw
Reset Value	0x00000000
Description	CoreSight Trace ID Register

Register ETMTRACEIDR Details

Field Name	Bits	Type	Reset Value	Description
TraceID	6:0	rw	0x0	Before trace is generated, you must program this register with a non-reserved value. Reserved values are 0x00 and any value in the range 0x70-0x7F. The reset value of this register is 0x00.

Register ([ptm](#)) OSLSR

Name	OSLSR
Relative Address	0x00000304
Absolute Address	debug_cpu_ptm0: 0xF889C304 debug_cpu_ptm1: 0xF889D304

Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	OS Lock Status Register

Register OSLSR Details

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	Shows that OS Locking is not implemented.

Register ([ptm](#)) ETMPDSR

Name	ETMPDSR
Relative Address	0x00000314
Absolute Address	debug_cpu_ptm0: 0xF889C314 debug_cpu_ptm1: 0xF889D314
Width	32 bits
Access Type	ro
Reset Value	0x00000001
Description	Device Powerdown Status Register

Register ETMPDSR Details

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x1	Indicates that the PTM Trace Registers can be accessed.

Register ([ptm](#)) ITMISCOUT

Name	ITMISCOUT
Relative Address	0x00000EDC
Absolute Address	debug_cpu_ptm0: 0xF889CEDC debug_cpu_ptm1: 0xF889DEDCE
Width	10 bits
Access Type	wo
Reset Value	0x00000000
Description	Miscellaneous Outputs Register

Register ITMISCOUT Details

Field Name	Bits	Type	Reset Value	Description
PTMEXTOUT	9:8	wo	0x0	Drives the PTMEXTOUT[1:0] outputs
reserved	7:6	wo	0x0	Reserved
PTMIDLEACK	5	wo	0x0	Drives the PTMIDLEACK output
PTMDBGREQ	4	wo	0x0	Drives the PTMDBGREQ output
reserved	3:0	wo	0x0	Reserved

Register ([ptm](#)) ITMISCIN

Name	ITMISCIN
Relative Address	0x00000EE0
Absolute Address	debug_cpu_ptm0: 0xF889CEE0 debug_cpu_ptm1: 0xF889DEE0
Width	7 bits
Access Type	ro
Reset Value	x
Description	Miscellaneous Inputs Register

Register ITMISCIN Details

Field Name	Bits	Type	Reset Value	Description
STANDBYWFI	6	ro	x	Returns the value of the STANDBYWFI input
reserved	5	ro	0x0	Reserved
PTMDBGACK	4	ro	x	Returns the value of the PTMDBGACK input
EXTIN	3:0	ro	x	Returns the value of the EXTIN[3:0] inputs

Register ([ptm](#)) ITTRIGGER

Name	ITTRIGGER
Relative Address	0x00000EE8
Absolute Address	debug_cpu_ptm0: 0xF889CEE8 debug_cpu_ptm1: 0xF889DEE8
Width	1 bits
Access Type	wo
Reset Value	0x00000000
Description	Trigger Register

Register ITTRIGGER Details

Field Name	Bits	Type	Reset Value	Description
PTMTRIGGER	0	wo	0x0	Drives the PTMTRIGGER output

Register ([ptm](#)) ITATBDATA0

Name	ITATBDATA0
Relative Address	0x0000EEC
Absolute Address	debug_cpu_ptm0: 0xF889CEEC debug_cpu_ptm1: 0xF889DEEC
Width	5 bits
Access Type	wo
Reset Value	0x00000000
Description	ATB Data 0 Register

Register ITATBDATA0 Details

Field Name	Bits	Type	Reset Value	Description
ATDATAM31	4	wo	0x0	Drives the ATDATAM[31] output
ATDATAM23	3	wo	0x0	Drives the ATDATAM[23] output
ATDATAM15	2	wo	0x0	Drives the ATDATAM[15] output
ATDATAM7	1	wo	0x0	Drives the ATDATAM[7] output
ATDATAM0	0	wo	0x0	Drives the ATDATAM[0] output

Register ([ptm](#)) ITATBCTR2

Name	ITATBCTR2
Relative Address	0x0000EF0
Absolute Address	debug_cpu_ptm0: 0xF889CEF0 debug_cpu_ptm1: 0xF889DEF0
Width	2 bits
Access Type	ro
Reset Value	x
Description	ATB Control 2 Register

Register ITATBCTR2 Details

Field Name	Bits	Type	Reset Value	Description
AFVALIDM	1	ro	x	Returns the value of the AFVALIDM input
ATREADYM	0	ro	x	Returns the value of the ATREADYM input

Register ([ptm](#)) ITATBID

Name	ITATBID
Relative Address	0x00000EF4
Absolute Address	debug_cpu_ptm0: 0xF889CEF4 debug_cpu_ptm1: 0xF889DEF4
Width	7 bits
Access Type	wo
Reset Value	0x00000000
Description	ATB Identification Register

Register ITATBID Details

Field Name	Bits	Type	Reset Value	Description
ATIDM	6:0	wo	0x0	Drives the ATIDM[6:0] outputs

Register ([ptm](#)) ITATBCTR0

Name	ITATBCTR0
Relative Address	0x00000EF8
Absolute Address	debug_cpu_ptm0: 0xF889CEF8 debug_cpu_ptm1: 0xF889DEF8
Width	10 bits
Access Type	wo
Reset Value	0x00000000
Description	ATB Control 0 Register

Register ITATBCTR0 Details

Field Name	Bits	Type	Reset Value	Description
ATBYTESM	9:8	wo	0x0	Drives the ATBYTESM[9:8] outputs
reserved	7:2	wo	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
AFREADYM	1	wo	0x0	Drives the AFREADYM output
ATVALIDM	0	wo	0x0	Drives the ATVALIDM output

Register ([ptm](#)) ETMITCTRL

Name	ETMITCTRL
Relative Address	0x00000F00
Absolute Address	debug_cpu_ptm0: 0xF889CF00 debug_cpu_ptm1: 0xF889DF00
Width	1 bits
Access Type	rw
Reset Value	0x00000000
Description	Integration Mode Control Register

Register ETMITCTRL Details

Field Name	Bits	Type	Reset Value	Description
	0	rw	0x0	Enable Integration Test registers. Before entering integration mode, the PTM must be powered up and in programming mode. THis means bit 0 of the Main Control Register is set to 0, and bit 10 of the Main Control Register ist set 1. After leaving integration mode, the PTM must be reset before attempting to perform tracing.

Register ([ptm](#)) CTSR

Name	CTSR
Relative Address	0x00000FA0
Absolute Address	debug_cpu_ptm0: 0xF889CFA0 debug_cpu_ptm1: 0xF889DFA0
Width	8 bits
Access Type	rw
Reset Value	0x000000FF
Description	Claim Tag Set Register

Register CTSR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	rw	0xFF	<p>The claim tag register is used for any interrogating tools to determine if the device is being programmed or has been programmed.</p> <p>Read: 1= Claim tag is implemented, 0 = Claim tag is not implemented</p> <p>Write: 1= Set claim tag bit, 0= No effect</p>

Register ([ptm](#)) CTCR

Name	CTCR
Relative Address	0x00000FA4
Absolute Address	debug_cpu_ptm0: 0xF889CFA4 debug_cpu_ptm1: 0xF889DFA4
Width	8 bits
Access Type	rw
Reset Value	0x00000000
Description	Claim Tag Clear Register

Register CTCR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	rw	0x0	<p>The claim tag register is used for any interrogating tools to determine if the device is being programmed or has been programmed.</p> <p>Read: Current value of claim tag.</p> <p>Write: 1= Clear claim tag bit, 0= No effect</p>

Register ([ptm](#)) LAR

Name	LAR
Relative Address	0x00000FB0
Absolute Address	debug_cpu_ptm0: 0xF889CFB0 debug_cpu_ptm1: 0xF889DFB0
Width	32 bits
Access Type	wo
Reset Value	0x00000000

Description Lock Access Register

Register LAR Details

Field Name	Bits	Type	Reset Value	Description
	31:0	wo	0x0	<p>Write Access Code.</p> <p>Write behavior depends on PADDRDBG31 pin:</p> <ul style="list-style-type: none"> - PADDRDBG31=0 (lower 2GB): <p>After reset (via PRESETDBGn), PTM is locked, i.e., writes to all other registers using lower 2GB addresses are ignored.</p> <p>To unlock, 0xC5ACCE55 must be written this register.</p> <p>After the required registers are written, to lock again, write a value other than 0xC5ACCE55 to this register.</p> <ul style="list-style-type: none"> - PADDRDBG31=1 (upper 2GB): <p>PTM is unlocked when upper 2GB addresses are used to write to all the registers.</p> <p>However, write to this register is ignored using a upper 2GB address!</p> <p>Note: read from this register always returns 0, regardless of PADDRDBG31.</p>

Register ([ptm](#)) LSR

Name	LSR
Relative Address	0x00000FB4
Absolute Address	debug_cpu_ptm0: 0xF889CFB4 debug_cpu_ptm1: 0xF889DFB4
Width	3 bits
Access Type	ro
Reset Value	0x00000003
Description	Lock Status Register

Register LSR Details

Field Name	Bits	Type	Reset Value	Description
8BIT	2	ro	0x0	Set to 0 since PTM implements a 32-bit lock access register
STATUS	1	ro	0x1	Read behavior depends on PADDRDBG31 pin: - PADDRDBG31=0 (lower 2GB): When a lower 2GB address is used to read this register, this bit indicates whether PTM is in locked state (1= locked, 0= unlocked). - PADDRDBG31=1 (upper 2GB): always returns 0.
IMP	0	ro	0x1	Read behavior depends on PADDRDBG31 pin: - PADDRDBG31=0 (lower 2GB): always returns 1, meaning lock mechanism are implemented. - PADDRDBG31=1 (upper 2GB): always returns 0, meaning lock mechanism is NOT implemented.

Register ([ptm](#)) ASR

Name	ASR
Relative Address	0x00000FB8
Absolute Address	debug_cpu_ptm0: 0xF889CFB8 debug_cpu_ptm1: 0xF889DFB8
Width	8 bits
Access Type	ro
Reset Value	x
Description	Authentication Status Register

Register ASR Details

Field Name	Bits	Type	Reset Value	Description
SNI	7:6	ro	0x0	Secure non-invasive debug Always 2'b00. This functionality is not implemented
SI	5:4	ro	0x0	Secure invasive debug Always 2'b00. This functionality is not implemented.

Field Name	Bits	Type	Reset Value	Description
NSNI	3:2	ro	x	Non-secure non-invasive debug IF NIDEN or DBGEN is 1, this field is 2'b11, indicating the functionality is implemented and enabled. Otherwise, this field is 2'b10 (implemented but disabled)
NSI	1:0	ro	0x0	Non-secure invasive debug Always 2'b00. This functionality is not implemented.

Register ([ptm](#)) DEVID

Name	DEVID
Relative Address	0x00000FC8
Absolute Address	debug_cpu_ptm0: 0xF889CFC8 debug_cpu_ptm1: 0xF889DFC8
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Device ID

Register DEVID Details

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	Component capability

Register ([ptm](#)) DTIR

Name	DTIR
Relative Address	0x00000FCC
Absolute Address	debug_cpu_ptm0: 0xF889CFCC debug_cpu_ptm1: 0xF889DFCC
Width	8 bits
Access Type	ro
Reset Value	0x00000013
Description	Device Type Identifier Register

Register DTIR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x13	A trace source and processor trace

Register ([ptm](#)) PERIPHD4

Name	PERIPHD4
Relative Address	0x0000FD0
Absolute Address	debug_cpu_ptm0: 0xF889CFD0 debug_cpu_ptm1: 0xF889DFD0
Width	8 bits
Access Type	ro
Reset Value	0x00000004
Description	Peripheral ID4

Register PERIPHD4 Details

Field Name	Bits	Type	Reset Value	Description
4KB_count	7:4	ro	0x0	4KB Count, set to 0
JEP106ID	3:0	ro	0x4	JEP106 continuation code

Register ([ptm](#)) PERIPHD5

Name	PERIPHD5
Relative Address	0x0000FD4
Absolute Address	debug_cpu_ptm0: 0xF889CFD4 debug_cpu_ptm1: 0xF889DFD4
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID5

Register PERIPHD5 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([ptm](#)) PERIPHID6

Name	PERIPHID6
Relative Address	0x0000FD8
Absolute Address	debug_cpu_ptm0: 0xF889CFD8 debug_cpu_ptm1: 0xF889DFD8
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID6

Register PERIPHID6 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([ptm](#)) PERIPHID7

Name	PERIPHID7
Relative Address	0x0000FDC
Absolute Address	debug_cpu_ptm0: 0xF889CFDC debug_cpu_ptm1: 0xF889DFDC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID7

Register PERIPHID7 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([ptm](#)) PERIPHID0

Name	PERIPHID0
Relative Address	0x0000FE0
Absolute Address	debug_cpu_ptm0: 0xF889CFE0 debug_cpu_ptm1: 0xF889DFE0
Width	8 bits

Access Type ro

Reset Value 0x00000050

Description Peripheral ID0

Register PERIPHID0 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x50	PartNumber0

Register ([ptm](#)) PERIPHID1

Name PERIPHID1

Relative Address 0x00000FE4

Absolute Address debug_cpu_ptm0: 0xF889CFE4
debug_cpu_ptm1: 0xF889DFE4

Width 8 bits

Access Type ro

Reset Value 0x000000B9

Description Peripheral ID1

Register PERIPHID1 Details

Field Name	Bits	Type	Reset Value	Description
JEP106ID	7:4	ro	0xB	JEP106 Identity Code [3:0]
PartNumber1	3:0	ro	0x9	PartNumber1

Register ([ptm](#)) PERIPHID2

Name PERIPHID2

Relative Address 0x00000FE8

Absolute Address debug_cpu_ptm0: 0xF889CFE8
debug_cpu_ptm1: 0xF889DFE8

Width 8 bits

Access Type ro

Reset Value 0x0000001B

Description Peripheral ID2

Register PERIPHD2 Details

Field Name	Bits	Type	Reset Value	Description
RevNum	7:4	ro	0x1	Revision number of Peripheral
JEDEC	3	ro	0x1	Indicates that a JEDEC assigned value is used
JEP106ID	2:0	ro	0x3	JEP106 Identity Code [6:4]

Register ([ptm](#)) PERIPHD3

Name	PERIPHD3
Relative Address	0x00000FEC
Absolute Address	debug_cpu_ptm0: 0xF889CFEC debug_cpu_ptm1: 0xF889DFEC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID3

Register PERIPHD3 Details

Field Name	Bits	Type	Reset Value	Description
RevAnd	7:4	ro	0x0	RevAnd, at top level
CustMod	3:0	ro	0x0	Customer Modified

Register ([ptm](#)) COMPID0

Name	COMPID0
Relative Address	0x00000FF0
Absolute Address	debug_cpu_ptm0: 0xF889CFF0 debug_cpu_ptm1: 0xF889DFF0
Width	8 bits
Access Type	ro
Reset Value	0x0000000D
Description	Component ID0

Register COMPID0 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0xD	Preamble

Register ([ptm](#)) COMPID1

Name	COMPID1
Relative Address	0x0000FF4
Absolute Address	debug_cpu_ptm0: 0xF889CFF4 debug_cpu_ptm1: 0xF889DFF4
Width	8 bits
Access Type	ro
Reset Value	0x00000090
Description	Component ID1

Register COMPID1 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x90	Preamble

Register ([ptm](#)) COMPID2

Name	COMPID2
Relative Address	0x0000FF8
Absolute Address	debug_cpu_ptm0: 0xF889CFF8 debug_cpu_ptm1: 0xF889DFF8
Width	8 bits
Access Type	ro
Reset Value	0x00000005
Description	Component ID2

Register COMPID2 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x5	Preamble

Register ([ptm](#)) COMPID3

Name	COMPID3
Relative Address	0x0000FFC
Absolute Address	debug_cpu_ptm0: 0xF889CFFC debug_cpu_ptm1: 0xF889DFFC
Width	8 bits

Access Type ro
Reset Value 0x000000B1
Description Component ID3

Register COMPID3 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0xB1	Preamble

B.10 Debug Access Port (dap)

Module Name	Debug Access Port (dap)
Base Address	0xF8800000 debug_dap_rom
Description	Debug Access Port ROM Table
Version	0.65
Doc Version	
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
ROMENTRY00	0x00000000	32	ro	0x00001003	ROM entry 00
ROMENTRY01	0x00000004	32	ro	0x00002003	ROM entry 01
ROMENTRY02	0x00000008	32	ro	0x00003003	ROM entry 02
ROMENTRY03	0x0000000C	32	ro	0x00004003	ROM entry 03
ROMENTRY04	0x00000010	32	ro	0x00005003	ROM entry 04
ROMENTRY05	0x00000014	32	ro	0x00009003	ROM entry 05
ROMENTRY06	0x00000018	32	ro	0x0000A003	ROM entry 06
ROMENTRY07	0x0000001C	32	ro	0x0000B003	ROM entry 07
ROMENTRY08	0x00000020	32	ro	0x0000C003	ROM entry 08
ROMENTRY09	0x00000024	32	ro	0x00080003	ROM entry 09
ROMENTRY10	0x00000028	32	rw	0x00000000	ROM entry 10
ROMENTRY11	0x0000002C	32	rw	0x00000000	ROM entry 11
ROMENTRY12	0x00000030	32	rw	0x00000000	ROM entry 12
ROMENTRY13	0x00000034	32	rw	0x00000000	ROM entry 13
ROMENTRY14	0x00000038	32	rw	0x00000000	ROM entry 14
ROMENTRY15	0x0000003C	32	rw	0x00000000	ROM entry 15
PERIPHID4	0x00000FD0	8	ro	0x00000003	Peripheral ID4
PERIPHID5	0x00000FD4	8	ro	0x00000000	Peripheral ID5
PERIPHID6	0x00000FD8	8	ro	0x00000000	Peripheral ID6
PERIPHID7	0x00000FDC	8	ro	0x00000000	Peripheral ID7
PERIPHID0	0x00000FE0	8	ro	0x000000B2	Peripheral ID0
PERIPHID1	0x00000FE4	8	ro	0x00000093	Peripheral ID1

Register Name	Address	Width	Type	Reset Value	Description
PERIPHID2	0x00000FE8	8	ro	0x00000008	Peripheral ID2
PERIPHID3	0x00000FEC	8	ro	0x00000000	Peripheral ID3
COMPID0	0x00000FF0	8	ro	0x0000000D	Component ID0
COMPID1	0x00000FF4	8	ro	0x00000010	Component ID1
COMPID2	0x00000FF8	8	ro	0x00000005	Component ID2
COMPID3	0x00000FFC	8	ro	0x000000B1	Component ID3

Register ([dap](#)) ROMENTRY00

Name	ROMENTRY00
Relative Address	0x00000000
Absolute Address	0xF8800000
Width	32 bits
Access Type	ro
Reset Value	0x00001003
Description	ROM entry 00

Register ROMENTRY00 Details

Field Name	Bits	Type	Reset Value	Description
AddressOffset	31:12	ro	0x1	Base address of the component, relative to the ROM address. Negative values are permitted using two's complement. ComponentAddress = ROMAddress + (AddressOffset SHL 12)
reserved	11:2	ro	0x0	Reserved
Format	1	ro	0x1	Format of ROM entry Enumerated Value List: 32BIT=1. 8BIT=0.
EntryPresent	0	ro	0x1	Set HIGH to indicate an entry is present.

Register ([dap](#)) ROMENTRY01

Name	ROMENTRY01
Relative Address	0x00000004
Absolute Address	0xF8800004

Width 32 bits

Access Type ro

Reset Value 0x00002003

Description ROM entry 01

Register ROMENTRY01 Details

Field Name	Bits	Type	Reset Value	Description
AddressOffset	31:12	ro	0x2	Base address of the component, relative to the ROM address. Negative values are permitted using two's complement. ComponentAddress = ROMAddress + (AddressOffset SHL 12)
reserved	11:2	ro	0x0	Reserved
Format	1	ro	0x1	Format of ROM entry Enumerated Value List: 32BIT=1. 8BIT=0.
EntryPresent	0	ro	0x1	Set HIGH to indicate an entry is present.

Register ([dap](#)) ROMENTRY02

Name ROMENTRY02

Relative Address 0x00000008

Absolute Address 0xF8800008

Width 32 bits

Access Type ro

Reset Value 0x00003003

Description ROM entry 02

Register ROMENTRY02 Details

Field Name	Bits	Type	Reset Value	Description
AddressOffset	31:12	ro	0x3	Base address of the component, relative to the ROM address. Negative values are permitted using two's complement. ComponentAddress = ROMAddress + (AddressOffset SHL 12)
reserved	11:2	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
Format	1	ro	0x1	Format of ROM entry Enumerated Value List: 32BIT=1. 8BIT=0.
EntryPresent	0	ro	0x1	Set HIGH to indicate an entry is present.

Register ([dap](#)) ROMENTRY03

Name	ROMENTRY03
Relative Address	0x0000000C
Absolute Address	0xF880000C
Width	32 bits
Access Type	ro
Reset Value	0x00004003
Description	ROM entry 03

Register ROMENTRY03 Details

Field Name	Bits	Type	Reset Value	Description
AddressOffset	31:12	ro	0x4	Base address of the component, relative to the ROM address. Negative values are permitted using two's complement. ComponentAddress = ROMAddress + (AddressOffset SHL 12)
reserved	11:2	ro	0x0	Reserved
Format	1	ro	0x1	Format of ROM entry Enumerated Value List: 32BIT=1. 8BIT=0.
EntryPresent	0	ro	0x1	Set HIGH to indicate an entry is present.

Register ([dap](#)) ROMENTRY04

Name	ROMENTRY04
Relative Address	0x00000010
Absolute Address	0xF8800010
Width	32 bits
Access Type	ro

Reset Value 0x00005003
Description ROM entry 04

Register ROMENTRY04 Details

Field Name	Bits	Type	Reset Value	Description
AddressOffset	31:12	ro	0x5	Base address of the component, relative to the ROM address. Negative values are permitted using two's complement. ComponentAddress = ROMAddress + (AddressOffset SHL 12)
reserved	11:2	ro	0x0	Reserved
Format	1	ro	0x1	Format of ROM entry Enumerated Value List: 32BIT=1. 8BIT=0.
EntryPresent	0	ro	0x1	Set HIGH to indicate an entry is present.

Register ([dap](#)) ROMENTRY05

Name ROMENTRY05
Relative Address 0x00000014
Absolute Address 0xF8800014
Width 32 bits
Access Type ro
Reset Value 0x00009003
Description ROM entry 05

Register ROMENTRY05 Details

Field Name	Bits	Type	Reset Value	Description
AddressOffset	31:12	ro	0x9	Base address of the component, relative to the ROM address. Negative values are permitted using two's complement. ComponentAddress = ROMAddress + (AddressOffset SHL 12)
reserved	11:2	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
Format	1	ro	0x1	Format of ROM entry Enumerated Value List: 32BIT=1. 8BIT=0.
EntryPresent	0	ro	0x1	Set HIGH to indicate an entry is present.

Register ([dap](#)) ROMENTRY06

Name	ROMENTRY06
Relative Address	0x00000018
Absolute Address	0xF8800018
Width	32 bits
Access Type	ro
Reset Value	0x0000A003
Description	ROM entry 06

Register ROMENTRY06 Details

Field Name	Bits	Type	Reset Value	Description
AddressOffset	31:12	ro	0xA	Base address of the component, relative to the ROM address. Negative values are permitted using two's complement. ComponentAddress = ROMAddress + (AddressOffset SHL 12)
reserved	11:2	ro	0x0	Reserved
Format	1	ro	0x1	Format of ROM entry Enumerated Value List: 32BIT=1. 8BIT=0.
EntryPresent	0	ro	0x1	Set HIGH to indicate an entry is present.

Register ([dap](#)) ROMENTRY07

Name	ROMENTRY07
Relative Address	0x0000001C
Absolute Address	0xF880001C
Width	32 bits
Access Type	ro

Reset Value 0x0000B003
Description ROM entry 07

Register ROMENTRY07 Details

Field Name	Bits	Type	Reset Value	Description
AddressOffset	31:12	ro	0xB	Base address of the component, relative to the ROM address. Negative values are permitted using two's complement. ComponentAddress = ROMAddress + (AddressOffset SHL 12)
reserved	11:2	ro	0x0	Reserved
Format	1	ro	0x1	Format of ROM entry Enumerated Value List: 32BIT=1. 8BIT=0.
EntryPresent	0	ro	0x1	Set HIGH to indicate an entry is present.

Register ([dap](#)) ROMENTRY08

Name ROMENTRY08
Relative Address 0x00000020
Absolute Address 0xF8800020
Width 32 bits
Access Type ro
Reset Value 0x0000C003
Description ROM entry 08

Register ROMENTRY08 Details

Field Name	Bits	Type	Reset Value	Description
AddressOffset	31:12	ro	0xC	Base address of the component, relative to the ROM address. Negative values are permitted using two's complement. ComponentAddress = ROMAddress + (AddressOffset SHL 12)
reserved	11:2	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
Format	1	ro	0x1	Format of ROM entry Enumerated Value List: 32BIT=1. 8BIT=0.
EntryPresent	0	ro	0x1	Set HIGH to indicate an entry is present.

Register ([dap](#)) ROMENTRY09

Name	ROMENTRY09
Relative Address	0x00000024
Absolute Address	0xF8800024
Width	32 bits
Access Type	ro
Reset Value	0x00080003
Description	ROM entry 09

Register ROMENTRY09 Details

Field Name	Bits	Type	Reset Value	Description
AddressOffset	31:12	ro	0x80	Base address of the component, relative to the ROM address. Negative values are permitted using two's complement. ComponentAddress = ROMAddress + (AddressOffset SHL 12)
reserved	11:2	ro	0x0	Reserved
Format	1	ro	0x1	Format of ROM entry Enumerated Value List: 32BIT=1. 8BIT=0.
EntryPresent	0	ro	0x1	Set HIGH to indicate an entry is present.

Register ([dap](#)) ROMENTRY10

Name	ROMENTRY10
Relative Address	0x00000028
Absolute Address	0xF8800028
Width	32 bits
Access Type	rw

Reset Value 0x00000000
Description ROM entry 10

Register ROMENTRY10 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Invalid entry

Register ([dap](#)) ROMENTRY11

Name ROMENTRY11
Relative Address 0x0000002C
Absolute Address 0xF880002C
Width 32 bits
Access Type rw
Reset Value 0x00000000
Description ROM entry 11

Register ROMENTRY11 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Invalid entry

Register ([dap](#)) ROMENTRY12

Name ROMENTRY12
Relative Address 0x00000030
Absolute Address 0xF8800030
Width 32 bits
Access Type rw
Reset Value 0x00000000
Description ROM entry 12

Register ROMENTRY12 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Invalid entry

Register ([dap](#)) ROMENTRY13

Name	ROMENTRY13
Relative Address	0x00000034
Absolute Address	0xF8800034
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	ROM entry 13

Register ROMENTRY13 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Invalid entry

Register ([dap](#)) ROMENTRY14

Name	ROMENTRY14
Relative Address	0x00000038
Absolute Address	0xF8800038
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	ROM entry 14

Register ROMENTRY14 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Invalid entry

Register ([dap](#)) ROMENTRY15

Name	ROMENTRY15
Relative Address	0x0000003C
Absolute Address	0xF880003C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	ROM entry 15

Register ROMENTRY15 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Invalid entry

Register ([dap](#)) PERIPHID4

Name	PERIPHID4
Relative Address	0x0000FD0
Absolute Address	0xF880FD0
Width	8 bits
Access Type	ro
Reset Value	0x00000003
Description	Peripheral ID4

Register PERIPHID4 Details

Field Name	Bits	Type	Reset Value	Description
4KB_count	7:4	ro	0x0	4KB Count, set to 0
	3:0	ro	0x3	JEP106 continuation code

Register ([dap](#)) PERIPHID5

Name	PERIPHID5
Relative Address	0x0000FD4
Absolute Address	0xF880FD4
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID5

Register PERIPHID5 Details

Field Name	Bits	Type	Reset Value	Description
reserved	7:0	ro	0x0	Reserved

Register ([dap](#)) PERIPHID6

Name	PERIPHID6
------	-----------

Relative Address	0x0000FD8
Absolute Address	0xF880FD8
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID6

Register PERIPHD6 Details

Field Name	Bits	Type	Reset Value	Description
reserved	7:0	ro	0x0	Reserved

Register ([dap](#)) PERIPHD7

Name	PERIPHD7
Relative Address	0x0000FDC
Absolute Address	0xF880FDC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID7

Register PERIPHD7 Details

Field Name	Bits	Type	Reset Value	Description
reserved	7:0	ro	0x0	Reserved

Register ([dap](#)) PERIPHD0

Name	PERIPHD0
Relative Address	0x0000FE0
Absolute Address	0xF880FE0
Width	8 bits
Access Type	ro
Reset Value	0x000000B2
Description	Peripheral ID0

Register PERIPHD0 Details

Field Name	Bits	Type	Reset Value	Description
PartNumber0	7:0	ro	0xB2	PartNumber0

Register ([dap](#)) PERIPHD1

Name	PERIPHD1
Relative Address	0x0000FE4
Absolute Address	0xF880FE4
Width	8 bits
Access Type	ro
Reset Value	0x0000093
Description	Peripheral ID1

Register PERIPHD1 Details

Field Name	Bits	Type	Reset Value	Description
JEP106ID	7:4	ro	0x9	JEP106 Identity Code [3:0]
PartNumber1	3:0	ro	0x3	PartNumber1

Register ([dap](#)) PERIPHD2

Name	PERIPHD2
Relative Address	0x0000FE8
Absolute Address	0xF880FE8
Width	8 bits
Access Type	ro
Reset Value	0x0000008
Description	Peripheral ID2

Register PERIPHD2 Details

Field Name	Bits	Type	Reset Value	Description
RevNum	7:4	ro	0x0	Revision number of Peripheral
JEDEC	3	ro	0x1	Indicates that a JEDEC assigned value is used
JEP106ID	2:0	ro	0x0	JEP106 Identity Code [6:4]

Register ([dap](#)) PERIPID3

Name	PERIPID3
Relative Address	0x0000FEC
Absolute Address	0xF8800FEC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID3

Register PERIPID3 Details

Field Name	Bits	Type	Reset Value	Description
RevAnd	7:4	ro	0x0	RevAnd, at top level
CustMod	3:0	ro	0x0	Customer Modified

Register ([dap](#)) COMPID0

Name	COMPID0
Relative Address	0x0000FF0
Absolute Address	0xF8800FF0
Width	8 bits
Access Type	ro
Reset Value	0x0000000D
Description	Component ID0

Register COMPID0 Details

Field Name	Bits	Type	Reset Value	Description
Preamble	7:0	ro	0xD	Preamble

Register ([dap](#)) COMPID1

Name	COMPID1
Relative Address	0x0000FF4
Absolute Address	0xF8800FF4
Width	8 bits
Access Type	ro
Reset Value	0x00000010

Description Component ID1

Register COMPID1 Details

Field Name	Bits	Type	Reset Value	Description
Preamble	7:0	ro	0x10	Preamble

Register ([dap](#)) COMPID2

Name COMPID2

Relative Address 0x0000FF8

Absolute Address 0xF880FF8

Width 8 bits

Access Type ro

Reset Value 0x00000005

Description Component ID2

Register COMPID2 Details

Field Name	Bits	Type	Reset Value	Description
Preamble	7:0	ro	0x5	Preamble

Register ([dap](#)) COMPID3

Name COMPID3

Relative Address 0x0000FFC

Absolute Address 0xF880FFC

Width 8 bits

Access Type ro

Reset Value 0x000000B1

Description Component ID3

Register COMPID3 Details

Field Name	Bits	Type	Reset Value	Description
Preamble	7:0	ro	0xB1	Preamble

B.11 CoreSight Embedded Trace Buffer (etb)

Module Name	CoreSight Embedded Trace Buffer (etb)
Base Address	0xF8801000 debug_etb
Description	Embedded Trace Buffer
Version	0.65
Doc Version	
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
RDP	0x00000004	32	ro	0x00000400	RAM Depth Register
STS	0x0000000C	4	ro	0x00000000	Status Register
RRD	0x00000010	32	ro	0x00000000	RAM Read Data Register
RRP	0x00000014	10	rw	0x00000000	RAM Read Pointer Register
RWP	0x00000018	10	rw	0x00000000	RAM Write Pointer Register
TRG	0x0000001C	10	rw	0x00000000	Trigger Counter Register
CTL	0x00000020	1	rw	0x00000000	Control Register
RWD	0x00000024	32	rw	0x00000000	RAM Write Data Register
FFSR	0x00000300	2	ro	0x00000000	Formatter and Flush Status Register
FFCR	0x00000304	14	mixed	0x00000200	Formatter and Flush Control Register
ITMISCOP0	0x00000EE0	2	wo	0x00000000	Integration Test Miscellaneous Output Register 0
ITTRFLINACK	0x00000EE4	2	wo	0x00000000	Integration Test Trigger In and Flush In Acknowledge Register
ITTRFLIN	0x00000EE8	2	wo	0x00000000	Integration Test Trigger In and Flush In Register
ITATBDATA0	0x00000EEC	5	ro	0x00000000	Integration Test ATB Data Register
ITATBCTR2	0x00000EF0	2	wo	0x00000000	Integration Test ATB Control Register 2
ITATBCTR1	0x00000EF4	7	ro	0x00000000	Integration Test ATB Control Register 1
ITATBCTR0	0x00000EF8	10	ro	0x00000000	Integration Test ATB Control Register 0

Register Name	Address	Width	Type	Reset Value	Description
IMCR	0x00000F00	1	rw	0x00000000	Integration Mode Control Register
CTSR	0x00000FA0	4	rw	0x0000000F	Claim Tag Set Register
CTCR	0x00000FA4	4	rw	0x00000000	Claim Tag Clear Register
LAR	0x00000FB0	32	wo	0x00000000	Lock Access Register
LSR	0x00000FB4	3	ro	0x00000003	Lock Status Register
ASR	0x00000FB8	8	ro	0x00000000	Authentication Status Register
DEVID	0x00000FC8	6	ro	0x00000000	Device ID
DTIR	0x00000FCC	8	ro	0x00000021	Device Type Identifier Register
PERIPID4	0x00000FD0	8	ro	0x00000004	Peripheral ID4
PERIPID5	0x00000FD4	8	ro	0x00000000	Peripheral ID5
PERIPID6	0x00000FD8	8	ro	0x00000000	Peripheral ID6
PERIPID7	0x00000FDC	8	ro	0x00000000	Peripheral ID7
PERIPID0	0x00000FE0	8	ro	0x00000007	Peripheral ID0
PERIPID1	0x00000FE4	8	ro	0x000000B9	Peripheral ID1
PERIPID2	0x00000FE8	8	ro	0x0000002B	Peripheral ID2
PERIPID3	0x00000FEC	8	ro	0x00000000	Peripheral ID3
COMPID0	0x00000FF0	8	ro	0x0000000D	Component ID0
COMPID1	0x00000FF4	8	ro	0x00000090	Component ID1
COMPID2	0x00000FF8	8	ro	0x00000005	Component ID2
COMPID3	0x00000FFC	8	ro	0x000000B1	Component ID3

Register ([etb](#)) RDP

Name	RDP
Relative Address	0x00000004
Absolute Address	0xF8801004
Width	32 bits
Access Type	ro
Reset Value	0x00000400
Description	RAM Depth Register

Register RDP Details

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x400	Defines the depth, in words, of the trace RAM.

Register ([etb](#)) STS

Name	STS
Relative Address	0x0000000C
Absolute Address	0xF880100C
Width	4 bits
Access Type	ro
Reset Value	0x00000000
Description	Status Register

Register STS Details

Field Name	Bits	Type	Reset Value	Description
FtEmpty	3	ro	0x0	Formatter pipeline empty. All data stored to RAM.
AcqComp	2	ro	0x0	Acquisition complete. The acquisition complete flag indicates that capture has been completed when the formatter stops because of any of the methods defined in the Formatter and Flush Control Register, or TraceCaptEn = 0. This also results in FtStopped in the Formatter and Flush Status Register going HIGH.
Triggered	1	ro	0x0	The Triggered bit is set when a trigger has been observed. This does not indicate that a trigger has been embedded in the trace data by the formatter, but is determined by the programming of the Formatter and Flush Control Register.
Full	0	ro	0x0	RAM Full. The flag indicates when the RAM write pointer has wrapped around.

Register ([etb](#)) RRD

Name	RRD
Relative Address	0x00000010
Absolute Address	0xF8801010
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	RAM Read Data Register

Register RRD Details

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	Data read from the ETB Trace RAM.

Register ([etb](#)) RRP

Name	RRP
Relative Address	0x00000014
Absolute Address	0xF8801014
Width	10 bits
Access Type	rw
Reset Value	0x00000000
Description	RAM Read Pointer Register

Register RRP Details

Field Name	Bits	Type	Reset Value	Description
	9:0	rw	0x0	Sets the read pointer used to read entries from the Trace RAM over the APB interface.

Register ([etb](#)) RWP

Name	RWP
Relative Address	0x00000018
Absolute Address	0xF8801018
Width	10 bits
Access Type	rw
Reset Value	0x00000000
Description	RAM Write Pointer Register

Register RWP Details

Field Name	Bits	Type	Reset Value	Description
	9:0	rw	0x0	Sets the write pointer used to write entries from the CoreSight bus into the Trace RAM

Register ([etb](#)) TRG

Name	TRG
------	-----

Relative Address	0x0000001C
Absolute Address	0xF880101C
Width	10 bits
Access Type	rw
Reset Value	0x00000000
Description	Trigger Counter Register

Register TRG Details

Field Name	Bits	Type	Reset Value	Description
	9:0	rw	0x0	<p>The counter is used as follows:</p> <ul style="list-style-type: none"> - Trace after <p>The counter is set to a large value, slightly less than the number of entries in the RAM.</p> <ul style="list-style-type: none"> - Trace before <p>The counter is set to a small value.</p> <ul style="list-style-type: none"> - Trace about <p>The counter is set to half the depth of the Trace RAM.</p> <p>This register must not be written to when trace capture is enabled (FtStopped=0, TraceCaptEn=1). If a write is attempted, the register is not updated. A read access is permitted with trace capture enabled.</p>

Register ([etb](#)) CTL

Name	CTL
Relative Address	0x00000020
Absolute Address	0xF8801020
Width	1 bits
Access Type	rw
Reset Value	0x00000000
Description	Control Register

Register CTL Details

Field Name	Bits	Type	Reset Value	Description
TraceCaptEn	0	rw	0x0	<p>ETB Trace Capture Enable.</p> <p>1 = enable trace capture</p> <p>0 = disable trace capture.</p> <p>This is the master enable bit forcing FtStopped HIGH when TraceCaptEn is LOW.</p> <p>When capture is disabled, any remaining data in the ATB formatter is stored to RAM.</p> <p>When all data is stored the formatter outputs FtStopped. Capture is fully disabled, or complete, when FtStopped goes HIGH.</p>

Register ([etb](#)) RWD

Name	RWD
Relative Address	0x00000024
Absolute Address	0xF8801024
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	RAM Write Data Register

Register RWD Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Data written to the ETB Trace RAM.</p> <p>When trace capture is disabled, the contents of this register are placed into the ETB Trace RAM when this register is written to.</p> <p>Writing to this register increments the RAM Write Pointer Register.</p> <p>If trace capture is enabled, and this register is accessed, then a read from this register outputs 0xFFFFFFFF. Reads of this register never increment the RAM Write Pointer Register. A constant stream of 1s being output corresponds to a synchronization output from the ETB. If a write access is attempted, the data is not written into Trace RAM.</p>

Register ([etb](#)) FFSR

Name	FFSR
------	------

Relative Address	0x00000300
Absolute Address	0xF8801300
Width	2 bits
Access Type	ro
Reset Value	0x00000000
Description	Formatter and Flush Status Register

Register FFSR Details

Field Name	Bits	Type	Reset Value	Description
FtStopped	1	ro	0x0	Formatter stopped. The formatter has received a stop request signal and all trace data and post-amble has been output. Any more trace data on the ATB interface is ignored and ATREADY goes HIGH.
FlInProg	0	ro	0x0	Flush In Progress. This is an indication of the current state of AFVALID.

Register (etb) FFCR

Name	FFCR
Relative Address	0x00000304
Absolute Address	0xF8801304
Width	14 bits
Access Type	mixed
Reset Value	0x00000200
Description	Formatter and Flush Control Register

Register FFCR Details

Field Name	Bits	Type	Reset Value	Description
StopTrig	13	rw	0x0	Stop the formatter when a Trigger Event has been observed.
StopFl	12	rw	0x0	Stop the formatter when a flush has completed (return of AFREADY). This forces the FIFO to drain off any part-completed packets. Setting this bit enables this function but this is clear on reset (disabled).
reserved	11	ro	0x0	Reserved
TrigFl	10	rw	0x0	Indicate a trigger on Flush completion (AFREADY being returned).
TrigEvt	9	rw	0x1	Indicate a trigger on a Trigger Event.

Field Name	Bits	Type	Reset Value	Description
TrigIn	8	rw	0x0	Indicate a trigger on TRIGIN being asserted.
reserved	7	ro	0x0	Reserved
FOnMan	6	rw	0x0	Manually generate a flush of the system. Setting this bit causes a flush to be generated. This is cleared when the flush has been serviced. This bit is clear on reset.
FOnTrig	5	rw	0x0	Generate flush using Trigger event. Set this bit to cause a flush of data in the system when a Trigger Event occurs. This bit is clear on reset.
FOnFlIn	4	rw	0x0	Generate flush using the FLUSHIN interface. Set this bit to enable use of the FLUSHIN connection. This bit is clear on reset.
reserved	3:2	ro	0x0	Reserved
EnFCont	1	rw	0x0	Continuous Formatting. Continuous mode in the ETB corresponds to normal mode with the embedding of triggers. Can only be changed when FtStopped is HIGH. This bit is clear on reset.
EnFTC	0	rw	0x0	Enable Formatting. Do not embed Triggers into the formatted stream. Trace disable cycles and triggers are indicated by TRACECTL, where fitted. Can only be changed when FtStopped is HIGH. This bit is clear on reset.

Register ([etb](#)) ITMISCOP0

Name	ITMISCOP0
Relative Address	0x0000EE0
Absolute Address	0xF8801EE0
Width	2 bits
Access Type	wo
Reset Value	0x00000000
Description	Integration Test Miscellaneous Output Register 0

Register ITMISCOP0 Details

Field Name	Bits	Type	Reset Value	Description
FULL	1	wo	0x0	Set the value of FULL
ACQCOMP	0	wo	0x0	Set the value of ACQCOMP

Register ([etb](#)) ITTRFLINACK

Name	ITTRFLINACK
Relative Address	0x0000EE4
Absolute Address	0xF8801EE4
Width	2 bits
Access Type	wo
Reset Value	0x00000000
Description	Integration Test Trigger In and Flush In Acknowledge Register

Register ITTRFLINACK Details

Field Name	Bits	Type	Reset Value	Description
FLUSHINACK	1	wo	0x0	Set the value of FLUSHINACK
TRIGINACK	0	wo	0x0	Set the value of TRIGINACK

Register ([etb](#)) ITTRFLIN

Name	ITTRFLIN
Relative Address	0x0000EE8
Absolute Address	0xF8801EE8
Width	2 bits
Access Type	wo
Reset Value	0x00000000
Description	Integration Test Trigger In and Flush In Register

Register ITTRFLIN Details

Field Name	Bits	Type	Reset Value	Description
FLUSHIN	1	wo	0x0	Read the value of FLUSHIN
TRIGIN	0	wo	0x0	Read the value of TRIGIN

Register ([etb](#)) ITATBDATA0

Name	ITATBDATA0
Relative Address	0x0000EEC
Absolute Address	0xF8801EEC
Width	5 bits
Access Type	ro

Reset Value 0x00000000

Description Integration Test ATB Data Register

Register ITATBDATA0 Details

Field Name	Bits	Type	Reset Value	Description
ATDATA31	4	ro	0x0	Read the value of ATDATA[31]
ATDATA23	3	ro	0x0	Read the value of ATDATA[23]
ATDATA15	2	ro	0x0	Read the value of ATDATA[15]
ATDATA7	1	ro	0x0	Read the value of ATDATA[7]
ATDATA0	0	ro	0x0	Read the value of ATDATA[0]

Register ([etb](#)) ITATBCTR2

Name ITATBCTR2

Relative Address 0x00000EF0

Absolute Address 0xF8801EF0

Width 2 bits

Access Type wo

Reset Value 0x00000000

Description Integration Test ATB Control Register 2

Register ITATBCTR2 Details

Field Name	Bits	Type	Reset Value	Description
AFVALIDS	1	wo	0x0	Set the value of AFVALIDS
ATREADYDYS	0	wo	0x0	Set the value of ATREADYDYS

Register ([etb](#)) ITATBCTR1

Name ITATBCTR1

Relative Address 0x00000EF4

Absolute Address 0xF8801EF4

Width 7 bits

Access Type ro

Reset Value 0x00000000

Description Integration Test ATB Control Register 1

Register ITATBCTR1 Details

Field Name	Bits	Type	Reset Value	Description
ATID	6:0	ro	0x0	Read the value of ATIDS

Register ([etb](#)) ITATBCTR0

Name	ITATBCTR0
Relative Address	0x00000EF8
Absolute Address	0xF8801EF8
Width	10 bits
Access Type	ro
Reset Value	0x00000000
Description	Integration Test ATB Control Register 0

Register ITATBCTR0 Details

Field Name	Bits	Type	Reset Value	Description
ATBYTES	9:8	ro	0x0	Read the value of ATBYTES
reserved	7:2	ro	0x0	Reserved
AFREADY	1	ro	0x0	Read the value of AFREADY
ATVALID	0	ro	0x0	Read the value of ATVALID

Register ([etb](#)) IMCR

Name	IMCR
Relative Address	0x00000F00
Absolute Address	0xF8801F00
Width	1 bits
Access Type	rw
Reset Value	0x00000000
Description	Integration Mode Control Register

Register IMCR Details

Field Name	Bits	Type	Reset Value	Description
	0	rw	0x0	Enable Integration Test registers.

Register ([etb](#)) CTSR

Name	CTSR
Relative Address	0x0000FA0
Absolute Address	0xF8801FA0
Width	4 bits
Access Type	rw
Reset Value	0x0000000F
Description	Claim Tag Set Register

Register CTSR Details

Field Name	Bits	Type	Reset Value	Description
	3:0	rw	0xF	The claim tag register is used for any interrogating tools to determine if the device is being programmed or has been programmed. Read: 1= Claim tag is implemented, 0 = Claim tag is not implemented Write: 1= Set claim tag bit, 0= No effect

Register ([etb](#)) CTCR

Name	CTCR
Relative Address	0x0000FA4
Absolute Address	0xF8801FA4
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	Claim Tag Clear Register

Register CTCR Details

Field Name	Bits	Type	Reset Value	Description
	3:0	rw	0x0	The claim tag register is used for any interrogating tools to determine if the device is being programmed or has been programmed. Read: Current value of claim tag. Write: 1= Clear claim tag bit, 0= No effect

Register ([etb](#)) LAR

Name	LAR
Relative Address	0x00000FB0
Absolute Address	0xF8801FB0
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	Lock Access Register

Register LAR Details

Field Name	Bits	Type	Reset Value	Description
	31:0	wo	0x0	<p>Write Access Code.</p> <p>Write behavior depends on PADDRDBG31 pin:</p> <ul style="list-style-type: none"> - PADDRDBG31=0 (lower 2GB): <p>After reset (via PRESETDBGn), ETB is locked, i.e., writes to all other registers using lower 2GB addresses are ignored.</p> <p>To unlock, 0xC5ACCE55 must be written this register.</p> <p>After the required registers are written, to lock again, write a value other than 0xC5ACCE55 to this register.</p> <ul style="list-style-type: none"> - PADDRDBG31=1 (upper 2GB): <p>ETB is unlocked when upper 2GB addresses are used to write to all the registers.</p> <p>However, write to this register is ignored using a upper 2GB address!</p> <p>Note: read from this register always returns 0, regardless of PADDRDBG31.</p>

Register ([etb](#)) LSR

Name	LSR
Relative Address	0x00000FB4
Absolute Address	0xF8801FB4
Width	3 bits
Access Type	ro
Reset Value	0x00000003
Description	Lock Status Register

Register LSR Details

Field Name	Bits	Type	Reset Value	Description
8BIT	2	ro	0x0	Set to 0 since ETB implements a 32-bit lock access register
STATUS	1	ro	0x1	Read behavior depends on PADDRDBG31 pin: - PADDRDBG31=0 (lower 2GB): When a lower 2GB address is used to read this register, this bit indicates whether ETB is in locked state (1= locked, 0= unlocked). - PADDRDBG31=1 (upper 2GB): always returns 0.
IMP	0	ro	0x1	Read behavior depends on PADDRDBG31 pin: - PADDRDBG31=0 (lower 2GB): always returns 1, meaning lock mechanism are implemented. - PADDRDBG31=1 (upper 2GB): always returns 0, meaning lock mechanism is NOT implemented.

Register (etb) ASR

Name	ASR
Relative Address	0x00000FB8
Absolute Address	0xF8801FB8
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Authentication Status Register

Register ASR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	Indicates functionality not implemented

Register (etb) DEVID

Name	DEVID
Relative Address	0x00000FC8
Absolute Address	0xF8801FC8
Width	6 bits

Access Type	ro
Reset Value	0x00000000
Description	Device ID

Register DEVID Details

Field Name	Bits	Type	Reset Value	Description
SyncATCLK	5	ro	0x0	ETB RAM is synchronous to ATCLK
InputMux	4:0	ro	0x0	no input multiplexing

Register ([etb](#)) DTIR

Name	DTIR
Relative Address	0x0000FCC
Absolute Address	0xF8801FCC
Width	8 bits
Access Type	ro
Reset Value	0x00000021
Description	Device Type Identifier Register

Register DTIR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x21	A trace sink and specifically an ETB

Register ([etb](#)) PERIPID4

Name	PERIPID4
Relative Address	0x0000FD0
Absolute Address	0xF8801FD0
Width	8 bits
Access Type	ro
Reset Value	0x00000004
Description	Peripheral ID4

Register PERIPHD4 Details

Field Name	Bits	Type	Reset Value	Description
4KB_count	7:4	ro	0x0	4KB Count, set to 0
JEP106ID	3:0	ro	0x4	JEP106 continuation code

Register ([etb](#)) PERIPHD5

Name	PERIPHD5
Relative Address	0x0000FD4
Absolute Address	0xF8801FD4
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID5

Register PERIPHD5 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([etb](#)) PERIPHD6

Name	PERIPHD6
Relative Address	0x0000FD8
Absolute Address	0xF8801FD8
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID6

Register PERIPHD6 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([etb](#)) PERIPHD7

Name	PERIPHD7
------	----------

Relative Address	0x0000FDC
Absolute Address	0xF8801FDC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID7

Register PERIPHD7 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([etb](#)) PERIPHD0

Name	PERIPHD0
Relative Address	0x0000FE0
Absolute Address	0xF8801FE0
Width	8 bits
Access Type	ro
Reset Value	0x00000007
Description	Peripheral ID0

Register PERIPHD0 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x7	PartNumber0

Register ([etb](#)) PERIPHD1

Name	PERIPHD1
Relative Address	0x0000FE4
Absolute Address	0xF8801FE4
Width	8 bits
Access Type	ro
Reset Value	0x000000B9
Description	Peripheral ID1

Register PERIPHD1 Details

Field Name	Bits	Type	Reset Value	Description
JEP106ID	7:4	ro	0xB	JEP106 Identity Code [3:0]
PartNumber1	3:0	ro	0x9	PartNumber1

Register ([etb](#)) PERIPHD2

Name	PERIPHD2
Relative Address	0x0000FE8
Absolute Address	0xF8801FE8
Width	8 bits
Access Type	ro
Reset Value	0x0000002B
Description	Peripheral ID2

Register PERIPHD2 Details

Field Name	Bits	Type	Reset Value	Description
RevNum	7:4	ro	0x2	Revision number of Peripheral
JEDEC	3	ro	0x1	Indicates that a JEDEC assigned value is used
JEP106ID	2:0	ro	0x3	JEP106 Identity Code [6:4]

Register ([etb](#)) PERIPHD3

Name	PERIPHD3
Relative Address	0x0000FEC
Absolute Address	0xF8801FEC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID3

Register PERIPHD3 Details

Field Name	Bits	Type	Reset Value	Description
RevAnd	7:4	ro	0x0	RevAnd, at top level
CustMod	3:0	ro	0x0	Customer Modified

Register ([etb](#)) COMPID0

Name	COMPID0
Relative Address	0x0000FF0
Absolute Address	0xF8801FF0
Width	8 bits
Access Type	ro
Reset Value	0x0000000D
Description	Component ID0

Register COMPID0 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0xD	Preamble

Register ([etb](#)) COMPID1

Name	COMPID1
Relative Address	0x0000FF4
Absolute Address	0xF8801FF4
Width	8 bits
Access Type	ro
Reset Value	0x00000090
Description	Component ID1

Register COMPID1 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x90	Preamble

Register ([etb](#)) COMPID2

Name	COMPID2
Relative Address	0x0000FF8
Absolute Address	0xF8801FF8
Width	8 bits
Access Type	ro
Reset Value	0x00000005
Description	Component ID2

Register COMPID2 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x5	Preamble

Register ([etb](#)) COMPID3

Name	COMPID3
Relative Address	0x0000FFC
Absolute Address	0xF8801FFC
Width	8 bits
Access Type	ro
Reset Value	0x000000B1
Description	Component ID3

Register COMPID3 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0xB1	Preamble

B.12 PL Fabric Trace Monitor (ftm)

Module Name	PL Fabric Trace Monitor (ftm)
Base Address	0xF880B000 debug_ftm
Description	Fabric Trace Macrocell
Version	0.65
Doc Version	
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
FTMGLBCTRL	0x00000000	1	rw	0x00000000	FTM Global Control Register
FTMSTATUS	0x00000004	8	ro	0x00000082	FTM Status Register
FTMCONTROL	0x00000008	3	rw	0x00000000	FTM Configuration
FTMP2FDBG0	0x0000000C	8	rw	0x00000000	FPGA Debug Register P2F0
FTMP2FDBG1	0x00000010	8	rw	0x00000000	FPGA Debug Register P2F1
FTMP2FDBG2	0x00000014	8	rw	0x00000000	FPGA Debug Register P2F2
FTMP2FDBG3	0x00000018	8	rw	0x00000000	FPGA Debug Register P2F3
FTMF2PDBG0	0x0000001C	8	ro	0x00000000	FPGA Debug Register F2P0
FTMF2PDBG1	0x00000020	8	ro	0x00000000	FPGA Debug Register F2P1
FTMF2PDBG2	0x00000024	8	ro	0x00000000	FPGA Debug Register F2P2
FTMF2PDBG3	0x00000028	8	ro	0x00000000	FPGA Debug Register F2P3
CYCOUNTPRE	0x0000002C	4	rw	0x00000000	AXI Cycle Count clock pre-scaler
FTMSYNCRELOAD	0x00000030	12	rw	0x00000000	FTM Synchronization Counter reload value
FTMSYNCCOUT	0x00000034	12	ro	0x00000000	FTM Synchronization Counter value
FTMATID	0x00000400	7	rw	0x00000000	FTM ATID Value Register
FTMITTRIGOUTACK	0x00000ED0	4	ro	0x00000000	Trigger Output Acknowledge Integration Test Register
FTMITTRIGGER	0x00000ED4	4	wo	0x00000000	Trigger Output Integration Test Register
FTMITTRACEDIS	0x00000ED8	1	ro	0x00000000	External Trace Disable Integration Test Register

Register Name	Address	Width	Type	Reset Value	Description
FTMITCYCCOUNT	0x00000EDC	32	rw	0x00000001	Cycle Counter Test Register
FTMITATBDATA0	0x00000EEC	5	wo	0x00000000	ATB Data Integration Test Register 0
FTMITATBCTR2	0x00000EF0	2	ro	0x00000001	ATB Control Integration Test Register 2
FTMITATBCTR1	0x00000EF4	7	rw	0x00000000	ATB Control Integration Test Register 1
FTMITATBCTR0	0x00000EF8	10	wo	0x00000000	ATB Control Integration Test Register 0
FTMITCR	0x00000F00	1	rw	0x00000000	FTM Test Control Register
CLAIMTAGSET	0x00000FA0	8	rw	0x000000FF	Claim Tag Set Register
CLAIMTAGCLR	0x00000FA4	8	rw	0x000000FF	Claim Tag Clear Register
LOCK_ACCESS	0x00000FB0	32	wo	0x00000000	Lock Access Register
LOCK_STATUS	0x00000FB4	3	ro	0x00000003	Lock Status Register
FTMAUTHSTATUS	0x00000FB8	8	ro	0x00000088	Authentication Status Register
FTMDEVID	0x00000FC8	1	ro	0x00000000	Device Configuration Register
FTMDEV_TYPE	0x00000FCC	8	ro	0x00000033	Device Type Identification Register
FTMPERIPHID4	0x00000FD0	8	ro	0x00000000	Peripheral ID4
FTMPERIPHID5	0x00000FD4	8	ro	0x00000000	Peripheral ID5
FTMPERIPHID6	0x00000FD8	8	ro	0x00000000	Peripheral ID6
FTMPERIPHID7	0x00000FDC	8	ro	0x00000000	Peripheral ID7
FTMPERIPHID0	0x00000FE0	8	ro	0x00000001	Peripheral ID0
FTMPERIPHID1	0x00000FE4	8	ro	0x00000090	Peripheral ID1
FTMPERIPHID2	0x00000FE8	8	ro	0x0000000C	Peripheral ID2
FTMPERIPHID3	0x00000FEC	8	ro	0x00000000	Peripheral ID3
FTMCOMPONID0	0x00000FF0	8	ro	0x0000000D	Component ID0
FTMCOMPONID1	0x00000FF4	8	ro	0x00000090	Component ID1
FTMCOMPONID2	0x00000FF8	8	ro	0x00000005	Component ID2
FTMCOMPONID3	0x00000FFC	8	ro	0x000000B1	Component ID3

Register ([ftm](#)) FTMGLBCTRL

Name	FTMGLBCTRL
Relative Address	0x00000000
Absolute Address	0xF880B000

Width	1 bits
Access Type	rw
Reset Value	0x00000000
Description	FTM Global Control Register

Register FTMGLBCTRL Details

Field Name	Bits	Type	Reset Value	Description
FTMENABLE	0	rw	0x0	Enable FTM

Register ([ftm](#)) FTMSTATUS

Name	FTMSTATUS
Relative Address	0x00000004
Absolute Address	0xF880B004
Width	8 bits
Access Type	ro
Reset Value	0x00000082
Description	FTM Status Register

Register FTMSTATUS Details

Field Name	Bits	Type	Reset Value	Description
IDLE	7	ro	0x1	FTM IDLE Status
SPIDEN	6	ro	0x0	Trustzone SPIDEN signal status
DBGGEN	5	ro	0x0	Trustzone DBGGEN signal status
SPNIDEN	4	ro	0x0	Trustzone SPNIDEN signal status
NIDEN	3	ro	0x0	Trustzone NIDEN signal status
FIFOFULL	2	ro	0x0	1 = FIFO is full
FIFOEMPTY	1	ro	0x1	1 = FIFO is empty
LOCKED	0	ro	0x0	Always read as zero

Register ([ftm](#)) FTMCONTROL

Name	FTMCONTROL
Relative Address	0x00000008
Absolute Address	0xF880B008
Width	3 bits

Access Type rw
 Reset Value 0x00000000
 Description FTM Configuration

Register FTMCONTROL Details

Field Name	Bits	Type	Reset Value	Description
CYCEN	2	rw	0x0	Enable Cycle Count packets
TRACEN	1	rw	0x0	Enable Trace packets
PROG	0	rw	0x0	Not used

Register ([ftm](#)) FTMP2FDBG0

Name FTMP2FDBG0
 Relative Address 0x0000000C
 Absolute Address 0xF880B00C
 Width 8 bits
 Access Type rw
 Reset Value 0x00000000
 Description FPGA Debug Register P2F0

Register FTMP2FDBG0 Details

Field Name	Bits	Type	Reset Value	Description
PSS2FPGA	7:0	rw	0x0	Signals presented to the fabric. These signals do not affect the FTM, they are provided for user specific debug. To modify the contents of this register, the SPIDEN pin must be asserted.

Register ([ftm](#)) FTMP2FDBG1

Name FTMP2FDBG1
 Relative Address 0x00000010
 Absolute Address 0xF880B010
 Width 8 bits
 Access Type rw
 Reset Value 0x00000000
 Description FPGA Debug Register P2F1

Register FTMP2FDBG1 Details

Field Name	Bits	Type	Reset Value	Description
PSS2FPGA	7:0	rw	0x0	Signals presented to the fabric. These signals do not affect the FTM, they are provided for user specific debug. To modify the contents of this register, the SPIDEN pin must be asserted.

Register ([ftm](#)) FTMP2FDBG2

Name	FTMP2FDBG2
Relative Address	0x00000014
Absolute Address	0xF880B014
Width	8 bits
Access Type	rw
Reset Value	0x00000000
Description	FPGA Debug Register P2F2

Register FTMP2FDBG2 Details

Field Name	Bits	Type	Reset Value	Description
PSS2FPGA	7:0	rw	0x0	Signals presented to the fabric. These signals do not affect the FTM, they are provided for user specific debug. To modify the contents of this register, the SPIDEN pin must be asserted.

Register ([ftm](#)) FTMP2FDBG3

Name	FTMP2FDBG3
Relative Address	0x00000018
Absolute Address	0xF880B018
Width	8 bits
Access Type	rw
Reset Value	0x00000000
Description	FPGA Debug Register P2F3

Register FTMP2FDBG3 Details

Field Name	Bits	Type	Reset Value	Description
PSS2FPGA	7:0	rw	0x0	Signals presented to the fabric. These signals do not affect the FTM, they are provided for user specific debug. To modify the contents of this register, the SPIDEN pin must be asserted.

Register ([ftm](#)) FTMF2PDBG0

Name	FTMF2PDBG0
Relative Address	0x0000001C
Absolute Address	0xF880B01C
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	FPGA Debug Register F2P0

Register FTMF2PDBG0 Details

Field Name	Bits	Type	Reset Value	Description
FPGA2PSS	7:0	ro	0x0	Signals that are presented to the PSS from the Fabric.

Register ([ftm](#)) FTMF2PDBG1

Name	FTMF2PDBG1
Relative Address	0x00000020
Absolute Address	0xF880B020
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	FPGA Debug Register F2P1

Register FTMF2PDBG1 Details

Field Name	Bits	Type	Reset Value	Description
FPGA2PSS	7:0	ro	0x0	Signals that are presented to the PSS from the Fabric.

Register ([ftm](#)) FTMF2PDBG2

Name	FTMF2PDBG2
Relative Address	0x00000024
Absolute Address	0xF880B024
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	FPGA Debug Register F2P2

Register FTMF2PDBG2 Details

Field Name	Bits	Type	Reset Value	Description
FPGA2PSS	7:0	ro	0x0	Signals that are presented to the PSS from the Fabric.

Register ([ftm](#)) FTMF2PDBG3

Name	FTMF2PDBG3
Relative Address	0x00000028
Absolute Address	0xF880B028
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	FPGA Debug Register F2P3

Register FTMF2PDBG3 Details

Field Name	Bits	Type	Reset Value	Description
FPGA2PSS	7:0	ro	0x0	Signals that are presented to the PSS from the Fabric.

Register ([ftm](#)) CYCOUNTPRE

Name	CYCOUNTPRE
Relative Address	0x0000002C
Absolute Address	0xF880B02C
Width	4 bits
Access Type	rw
Reset Value	0x00000000

Description AXI Cycle Count clock pre-scaler

Register CYCOUNTPRE Details

Field Name	Bits	Type	Reset Value	Description
PRESALE	3:0	rw	0x0	The incoming clock is divided by 2^{PRESALE} . For example: PRESALE = 15 indicates that the Cycle Counter runs at the AXI clock divided by $2^{15} = 32,768$ (PRESALE = 0 indicates no clock scaling)

Register ([ftm](#)) FTMSYNCRELOAD

Name FTMSYNCRELOAD

Relative Address 0x00000030

Absolute Address 0xF880B030

Width 12 bits

Access Type rw

Reset Value 0x00000000

Description FTM Synchronization Counter reload value

Register FTMSYNCRELOAD Details

Field Name	Bits	Type	Reset Value	Description
SYNCCOUNTTERM	11:0	rw	0x0	Reset FTM Synchronization packet counter when this number of packets has been transmitted. THIS NUMBER HAS A MINIMUM VALUE of 12.

Register ([ftm](#)) FTMSYNCCOUT

Name FTMSYNCCOUT

Relative Address 0x00000034

Absolute Address 0xF880B034

Width 12 bits

Access Type ro

Reset Value 0x00000000

Description FTM Synchronization Counter value

Register FTMSYNCCOUT Details

Field Name	Bits	Type	Reset Value	Description
SYNCCOUT	11:0	ro	0x0	Current value of the Synchronization packet counter. The initial value is zero. The counter value increments every time a packet is issued by the FTM. When the counter reaches SYNCCOUNTTERM, a Synchronization packet is emitted.

Register ([ftm](#)) FTMATID

Name	FTMATID
Relative Address	0x00000400
Absolute Address	0xF880B400
Width	7 bits
Access Type	rw
Reset Value	0x00000000
Description	FTM ATID Value Register

Register FTMATID Details

Field Name	Bits	Type	Reset Value	Description
ATID	6:0	rw	0x0	ATID value supplied to ATB bus. The upper three bits, ATID[6:4], are directly driven from this register. The lower four bits, ATID[3:0], are OR-ed with the FPGAATID[3:0] pins.

Register ([ftm](#)) FTMITTRIGOUTACK

Name	FTMITTRIGOUTACK
Relative Address	0x00000ED0
Absolute Address	0xF880BED0
Width	4 bits
Access Type	ro
Reset Value	0x00000000
Description	Trigger Output Acknowledge Integration Test Register

Register FTMITTRIGOUTACK Details

Field Name	Bits	Type	Reset Value	Description
TRIGACK	3:0	ro	0x0	Read the current value of the FTMTriGOutAck[3:0] inputs

Register ([ftm](#)) FTMITTRIGGER

Name	FTMITTRIGGER
Relative Address	0x00000ED4
Absolute Address	0xF880BED4
Width	4 bits
Access Type	wo
Reset Value	0x00000000
Description	Trigger Output Integration Test Register

Register FTMITTRIGGER Details

Field Name	Bits	Type	Reset Value	Description
TRIGGER	3:0	wo	0x0	When ITEN is 1, this field determines the FTMTriGOut[3:0]

Register ([ftm](#)) FTMITTRACEDIS

Name	FTMITTRACEDIS
Relative Address	0x00000ED8
Absolute Address	0xF880BED8
Width	1 bits
Access Type	ro
Reset Value	0x00000000
Description	External Trace Disable Integration Test Register

Register FTMITTRACEDIS Details

Field Name	Bits	Type	Reset Value	Description
TRACEDIS	0	ro	0x0	Always read as zero.

Register ([ftm](#)) FTMITCYCCOUNT

Name	FTMITCYCCOUNT
------	---------------

Relative Address	0x00000EDC
Absolute Address	0xF880BEDC
Width	32 bits
Access Type	rw
Reset Value	0x00000001
Description	Cycle Counter Test Register

Register FTMITCYCCOUNT Details

Field Name	Bits	Type	Reset Value	Description
FTMCYCCOUNT	31:0	rw	0x1	Read/write the value of the cycle counter

Register ([ftm](#)) FTMITATBDATA0

Name	FTMITATBDATA0
Relative Address	0x00000EEC
Absolute Address	0xF880BEEC
Width	5 bits
Access Type	wo
Reset Value	0x00000000
Description	ATB Data Integration Test Register 0

Register FTMITATBDATA0 Details

Field Name	Bits	Type	Reset Value	Description
ATDATA31	4	wo	0x0	When ITEN is 1, this value determines the ATDATAM[31] output
ATDATA23	3	wo	0x0	When ITEN is 1, this value determines the ATDATAM[23] output
ATDATA15	2	wo	0x0	When ITEN is 1, this value determines the ATDATAM[15] output
ATDATA7	1	wo	0x0	When ITEN is 1, this value determines the ATDATAM[7] output
ATDATA0	0	wo	0x0	When ITEN is 1, this value determines the ATDATAM[0] output

Register ([ftm](#)) FTMITATBCTR2

Name	FTMITATBCTR2
Relative Address	0x00000EF0

Absolute Address	0xF880BEF0
Width	2 bits
Access Type	ro
Reset Value	0x00000001
Description	ATB Control Integration Test Register 2

Register FTMITATBCTR2 Details

Field Name	Bits	Type	Reset Value	Description
AFVALID	1	ro	0x0	Read the current value of the AFVALIDM input
ATREADY	0	ro	0x1	Read the current value of the ATREADYM input

Register ([ftm](#)) FTMITATBCTR1

Name	FTMITATBCTR1
Relative Address	0x00000EF4
Absolute Address	0xF880BEF4
Width	7 bits
Access Type	rw
Reset Value	0x00000000
Description	ATB Control Integration Test Register 1

Register FTMITATBCTR1 Details

Field Name	Bits	Type	Reset Value	Description
ATID_test	6:0	rw	0x0	When ITEN is 1, this value determines the ATID output

Register ([ftm](#)) FTMITATBCTR0

Name	FTMITATBCTR0
Relative Address	0x00000EF8
Absolute Address	0xF880BEF8
Width	10 bits
Access Type	wo
Reset Value	0x00000000
Description	ATB Control Integration Test Register 0

Register FTMITATBCTR0 Details

Field Name	Bits	Type	Reset Value	Description
ATBYTES	9:8	wo	0x0	When ITEN is 1, this value determines the ATBYTESM[1:0] output
reserved	7:2	wo	0x0	Reserved
AFREADY	1	wo	0x0	When ITEN is 1, this value determines the AFREADY output
ATVALID	0	wo	0x0	When ITEN is 1, this value determines the ATVALID output

Register ([ftm](#)) FTMITCR

Name	FTMITCR
Relative Address	0x00000F00
Absolute Address	0xF880BF00
Width	1 bits
Access Type	rw
Reset Value	0x00000000
Description	FTM Test Control Register

Register FTMITCR Details

Field Name	Bits	Type	Reset Value	Description
ITEN	0	rw	0x0	Integration Test Enable

Register ([ftm](#)) CLAIMTAGSET

Name	CLAIMTAGSET
Relative Address	0x00000FA0
Absolute Address	0xF880BFA0
Width	8 bits
Access Type	rw
Reset Value	0x000000FF
Description	Claim Tag Set Register

Register CLAIMTAGSET Details

Field Name	Bits	Type	Reset Value	Description
CLAIMTAGSETVAL	7:0	rw	0xFF	Read: 1 = Claim tag implemented, 0 = not implemented Write: 1 = Set claim tag bit, 0 = no effect

Register ([ftm](#)) CLAIMTAGCLR

Name	CLAIMTAGCLR
Relative Address	0x0000FA4
Absolute Address	0xF880BFA4
Width	8 bits
Access Type	rw
Reset Value	0x000000FF
Description	Claim Tag Clear Register

Register CLAIMTAGCLR Details

Field Name	Bits	Type	Reset Value	Description
CLAIMTAGCLRVAL	7:0	rw	0xFF	Read: value of CLAIMTAGSETVAL Write: 1 = Clear claim tag bit, 0 = no effect

Register ([ftm](#)) LOCK_ACCESS

Name	LOCK_ACCESS
Relative Address	0x0000FB0
Absolute Address	0xF880BFB0
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	Lock Access Register

Register LOCK_ACCESS Details

Field Name	Bits	Type	Reset Value	Description
LOCKACCESS	31:0	wo	0x0	A value of 0xC5ACCE55 allows write access to FTM, any other value blocks write access

Register ([ftm](#)) LOCK_STATUS

Name	LOCK_STATUS
Relative Address	0x0000FB4
Absolute Address	0xF880BFB4
Width	3 bits
Access Type	ro
Reset Value	0x00000003
Description	Lock Status Register

Register LOCK_STATUS Details

Field Name	Bits	Type	Reset Value	Description
8BITACCESS	2	ro	0x0	8-bit lock access is not used
LOCKSTATUS	1	ro	0x1	1 = Access Locked, 0 = Access OK
LOCKIMP	0	ro	0x1	1 = Lock exists if PADDRDBG31 is low, else 0

Register ([ftm](#)) FTMAUTHSTATUS

Name	FTMAUTHSTATUS
Relative Address	0x0000FB8
Absolute Address	0xF880BFB8
Width	8 bits
Access Type	ro
Reset Value	0x00000088
Description	Authentication Status Register

Register FTMAUTHSTATUS Details

Field Name	Bits	Type	Reset Value	Description
AUTH_SPNIDEN	7:6	ro	0x2	Secure Non-Invasive Debug
reserved	5:4	ro	0x0	Secure Invasive Debug
AUTH_NIDEN	3:2	ro	0x2	Non-Secure Non-Invasive Debug
reserved	1:0	ro	0x0	Non-Secure Invasive Debug

Register ([ftm](#)) FTMDEVID

Name	FTMDEVID
Relative Address	0x0000FC8

Absolute Address	0xF880BFC8
Width	1 bits
Access Type	ro
Reset Value	0x00000000
Description	Device Configuration Register

Register FTMDEVID Details

Field Name	Bits	Type	Reset Value	Description
reserved	0	ro	0x0	Reserved

Register ([ftm](#)) FTMDEV_TYPE

Name	FTMDEV_TYPE
Relative Address	0x0000FCC
Absolute Address	0xF880BFCC
Width	8 bits
Access Type	ro
Reset Value	0x00000033
Description	Device Type Identification Register

Register FTMDEV_TYPE Details

Field Name	Bits	Type	Reset Value	Description
SubType	7:4	ro	0x3	Sub Type: Associated with a Data Engine or Co-processor
MajorType	3:0	ro	0x3	Major Type: Trace Source

Register ([ftm](#)) FTMPERIPHID4

Name	FTMPERIPHID4
Relative Address	0x0000FD0
Absolute Address	0xF880BFD0
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID4

Register FTMPERIPHID4 Details

Field Name	Bits	Type	Reset Value	Description
4KBCount	7:4	ro	0x0	4KB Count
JEP106	3:0	ro	0x0	JEP106 Continuation Code

Register ([ftm](#)) FTMPERIPHID5

Name	FTMPERIPHID5
Relative Address	0x0000FD4
Absolute Address	0xF880BFD4
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID5

Register FTMPERIPHID5 Details

Field Name	Bits	Type	Reset Value	Description
reserved	7:0	ro	0x0	Reserved

Register ([ftm](#)) FTMPERIPHID6

Name	FTMPERIPHID6
Relative Address	0x0000FD8
Absolute Address	0xF880BFD8
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID6

Register FTMPERIPHID6 Details

Field Name	Bits	Type	Reset Value	Description
reserved	7:0	ro	0x0	Reserved

Register ([ftm](#)) FTMPERIPHID7

Name	FTMPERIPHID7
------	--------------

Relative Address	0x0000FDC
Absolute Address	0xF880BFDC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID7

Register FTMPERIPHID7 Details

Field Name	Bits	Type	Reset Value	Description
reserved	7:0	ro	0x0	Reserved

Register ([ftm](#)) FTMPERIPHID0

Name	FTMPERIPHID0
Relative Address	0x0000FE0
Absolute Address	0xF880BFE0
Width	8 bits
Access Type	ro
Reset Value	0x00000001
Description	Peripheral ID0

Register FTMPERIPHID0 Details

Field Name	Bits	Type	Reset Value	Description
PARTNUMLOWER	7:0	ro	0x1	Part Number Lower

Register ([ftm](#)) FTMPERIPHID1

Name	FTMPERIPHID1
Relative Address	0x0000FE4
Absolute Address	0xF880BFE4
Width	8 bits
Access Type	ro
Reset Value	0x00000090
Description	Peripheral ID1

Register FTMPERIPHID1 Details

Field Name	Bits	Type	Reset Value	Description
JEP106	7:4	ro	0x9	JEP106 identity bits [3:0]
PARTNUMUPPER	3:0	ro	0x0	Part Number Upper [11:8]

Register ([ftm](#)) FTMPERIPHID2

Name	FTMPERIPHID2
Relative Address	0x00000FE8
Absolute Address	0xF880BFE8
Width	8 bits
Access Type	ro
Reset Value	0x0000000C
Description	Peripheral ID2

Register FTMPERIPHID2 Details

Field Name	Bits	Type	Reset Value	Description
REVISION	7:4	ro	0x0	Revision
JEDEC	3	ro	0x1	JEDEC used
JEP106	2:0	ro	0x4	JEP106 Identity [6:4]

Register ([ftm](#)) FTMPERIPHID3

Name	FTMPERIPHID3
Relative Address	0x00000FEC
Absolute Address	0xF880BFEC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID3

Register FTMPERIPHID3 Details

Field Name	Bits	Type	Reset Value	Description
RevAnd	7:4	ro	0x0	RevAnd
CustMod	3:0	ro	0x0	Customer Modified

Register ([ftm](#)) FTMCOMPONID0

Name	FTMCOMPONID0
Relative Address	0x0000FF0
Absolute Address	0xF880BFF0
Width	8 bits
Access Type	ro
Reset Value	0x0000000D
Description	Component ID0

Register FTMCOMPONID0 Details

Field Name	Bits	Type	Reset Value	Description
Preamble	7:0	ro	0xD	Preamble

Register ([ftm](#)) FTMCOMPONID1

Name	FTMCOMPONID1
Relative Address	0x0000FF4
Absolute Address	0xF880BFF4
Width	8 bits
Access Type	ro
Reset Value	0x00000090
Description	Component ID1

Register FTMCOMPONID1 Details

Field Name	Bits	Type	Reset Value	Description
CompClass	7:4	ro	0x9	Component Class = CoreSight Component
Preamble	3:0	ro	0x0	Preamble

Register ([ftm](#)) FTMCOMPONID2

Name	FTMCOMPONID2
Relative Address	0x0000FF8
Absolute Address	0xF880BFF8
Width	8 bits
Access Type	ro
Reset Value	0x00000005

Description Component ID2

Register FTMCOMPONID2 Details

Field Name	Bits	Type	Reset Value	Description
Preamble	7:0	ro	0x5	Preamble

Register ([ftm](#)) FTMCOMPONID3

Name FTMCOMPONID3

Relative Address 0x00000FFC

Absolute Address 0xF880BFFC

Width 8 bits

Access Type ro

Reset Value 0x000000B1

Description Component ID3

Register FTMCOMPONID3 Details

Field Name	Bits	Type	Reset Value	Description
Preamble	7:0	ro	0xB1	Preamble

B.13 CoreSight Trace Funnel (funnel)

Module Name	CoreSight Trace Funnel (funnel)
Base Address	0xF8804000 debug_funnel
Description	CoreSight Trace Funnel
Version	0.65
Doc Version	
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
Control	0x00000000	12	rw	0x00000300	CSTF Control Register
PriControl	0x00000004	24	rw	0x00FAC688	CSTF Priority Control Register
ITATBDATA0	0x00000EEC	5	rw	0x00000000	Integration Test ATB Data 0 Register
ITATBCTR2	0x00000EF0	2	rw	0x00000000	Integration Test ATB Control 2 Register
ITATBCTR1	0x00000EF4	7	rw	0x00000000	Integration Test ATB Control 1 Register
ITATBCTR0	0x00000EF8	10	mixed	0x00000000	Integration Test ATB Control 0 Register
IMCR	0x00000F00	1	rw	0x00000000	Integration Mode Control Register
CTSR	0x00000FA0	4	rw	0x0000000F	Claim Tag Set Register
CTCR	0x00000FA4	4	rw	0x00000000	Claim Tag Clear Register
LAR	0x00000FB0	32	wo	0x00000000	Lock Access Register
LSR	0x00000FB4	3	ro	0x00000003	Lock Status Register
ASR	0x00000FB8	8	ro	0x00000000	Authentication Status Register
DEVID	0x00000FC8	8	ro	0x00000028	Device ID
DTIR	0x00000FCC	8	ro	0x00000012	Device Type Identifier Register
PERIPID4	0x00000FD0	8	ro	0x00000004	Peripheral ID4
PERIPID5	0x00000FD4	8	ro	0x00000000	Peripheral ID5
PERIPID6	0x00000FD8	8	ro	0x00000000	Peripheral ID6
PERIPID7	0x00000FDC	8	ro	0x00000000	Peripheral ID7
PERIPID0	0x00000FE0	8	ro	0x00000008	Peripheral ID0

Register Name	Address	Width	Type	Reset Value	Description
PERIPID1	0x00000FE4	8	ro	0x000000B9	Peripheral ID1
PERIPID2	0x00000FE8	8	ro	0x0000001B	Peripheral ID2
PERIPID3	0x00000FEC	8	ro	0x00000000	Peripheral ID3
COMPID0	0x00000FF0	8	ro	0x0000000D	Component ID0
COMPID1	0x00000FF4	8	ro	0x00000090	Component ID1
COMPID2	0x00000FF8	8	ro	0x00000005	Component ID2
COMPID3	0x00000FFC	8	ro	0x000000B1	Component ID3

Register ([funnel](#)) Control

Name	Control
Relative Address	0x00000000
Absolute Address	0xF8804000
Width	12 bits
Access Type	rw
Reset Value	0x00000300
Description	CSTF Control Register

Register Control Details

Field Name	Bits	Type	Reset Value	Description
MinHoldTime	11:8	rw	0x3	The formatting scheme can easily become inefficient if fast switching occurs, so, where possible, this must be minimized. If a source has nothing to transmit, then another source is selected irrespective of the minimum number of cycles. Reset is 0x3. The CSTF holds for the minimum hold time and one additional cycle. The mFunnelum value that can be entered is 0xE and this equates to 15 cycles. 0xF is reserved.
EnableSlave7	7	rw	0x0	Setting this bit enables this slave port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme.
EnableSlave6	6	rw	0x0	Setting this bit enables this slave port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme.
EnableSlave5	5	rw	0x0	Setting this bit enables this slave port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme.

Field Name	Bits	Type	Reset Value	Description
EnableSlave4	4	rw	0x0	Setting this bit enables this slave port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme.
EnableSlave3	3	rw	0x0	Setting this bit enables this slave port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme.
EnableSlave2	2	rw	0x0	Setting this bit enables this slave port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme.
EnableSlave1	1	rw	0x0	Setting this bit enables this slave port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme.
EnableSlave0	0	rw	0x0	Setting this bit enables this slave port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme.

Register ([funnel](#)) PriControl

Name	PriControl
Relative Address	0x00000004
Absolute Address	0xF8804004
Width	24 bits
Access Type	rw
Reset Value	0x00FAC688
Description	CSTF Priority Control Register

Register PriControl Details

Field Name	Bits	Type	Reset Value	Description
PriPort7	23:21	rw	0x7	8th port priority value.
PriPort6	20:18	rw	0x6	7th port priority value.
PriPort5	17:15	rw	0x5	6th port priority value.
PriPort4	14:12	rw	0x4	5th port priority value.
PriPort3	11:9	rw	0x3	4th port priority value.
PriPort2	8:6	rw	0x2	3rd port priority value.
PriPort1	5:3	rw	0x1	2nd port priority value.
PriPort0	2:0	rw	0x0	1st port priority value.

Register ([funnel](#)) ITATBDATA0

Name	ITATBDATA0
Relative Address	0x00000EEC
Absolute Address	0xF8804EEC
Width	5 bits
Access Type	rw
Reset Value	0x00000000
Description	Integration Test ATB Data 0 Register

Register ITATBDATA0 Details

Field Name	Bits	Type	Reset Value	Description
ATDATA31	4	rw	0x0	Read the value of ATDATAS[31], set the value of ATDATAM[31]
ATDATA23	3	rw	0x0	Read the value of ATDATAS[23], set the value of ATDATAM[23]
ATDATA15	2	rw	0x0	Read the value of ATDATAS[15], set the value of ATDATAM[15]
ATDATA7	1	rw	0x0	Read the value of ATDATAS[7], set the value of ATDATAM[7]
ATDATA0	0	rw	0x0	Read the value of ATDATAS[0], set the value of ATDATAM[0]

Register ([funnel](#)) ITATBCTR2

Name	ITATBCTR2
Relative Address	0x00000EF0
Absolute Address	0xF8804EF0
Width	2 bits
Access Type	rw
Reset Value	0x00000000
Description	Integration Test ATB Control 2 Register

Register ITATBCTR2 Details

Field Name	Bits	Type	Reset Value	Description
AFREADY	1	rw	0x0	Read the value of AFVALIDM. Set the value of AFVALIDS<n>, where <n> is defined by the status of the CSTF Control Register.
	0	rw	0x0	Read the value of ATREADYM. Set the value of ATREADYDS<n>, where <n> is defined by the status of the CSTF Control Register.

Register ([funnel](#)) ITATBCTR1

Name	ITATBCTR1
Relative Address	0x00000EF4
Absolute Address	0xF8804EF4
Width	7 bits
Access Type	rw
Reset Value	0x00000000
Description	Integration Test ATB Control 1 Register

Register ITATBCTR1 Details

Field Name	Bits	Type	Reset Value	Description
ATID	6:0	rw	0x0	Read the value of ATIDS. Set the value of ATIDM.

Register ([funnel](#)) ITATBCTR0

Name	ITATBCTR0
Relative Address	0x00000EF8
Absolute Address	0xF8804EF8
Width	10 bits
Access Type	mixed
Reset Value	0x00000000
Description	Integration Test ATB Control 0 Register

Register ITATBCTR0 Details

Field Name	Bits	Type	Reset Value	Description
ATBYTES	9:8	rw	0x0	Read the value of ATBYTESS<n>. Set the value of ATBYTESM.
reserved	7:2	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
AFREADY	1	rw	0x0	Read the value of AFREADY<n>. Set the value of AFREADYM.
ATVALID	0	rw	0x0	Read the value of ATVALID<n>. Set the value of ATVALIDM.

Register ([funnel](#)) IMCR

Name	IMCR
Relative Address	0x00000F00
Absolute Address	0xF8804F00
Width	1 bits
Access Type	rw
Reset Value	0x00000000
Description	Integration Mode Control Register

Register IMCR Details

Field Name	Bits	Type	Reset Value	Description
	0	rw	0x0	Enable Integration Test registers.

Register ([funnel](#)) CTSR

Name	CTSR
Relative Address	0x00000FA0
Absolute Address	0xF8804FA0
Width	4 bits
Access Type	rw
Reset Value	0x0000000F
Description	Claim Tag Set Register

Register CTSR Details

Field Name	Bits	Type	Reset Value	Description
	3:0	rw	0xF	The claim tag register is used for any interrogating tools to determine if the device is being programmed or has been programmed. Read: 1= Claim tag is implemented, 0 = Claim tag is not implemented Write: 1= Set claim tag bit, 0= No effect

Register ([funnel](#)) CTCR

Name	CTCR
Relative Address	0x0000FA4
Absolute Address	0xF8804FA4
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	Claim Tag Clear Register

Register CTCR Details

Field Name	Bits	Type	Reset Value	Description
	3:0	rw	0x0	The claim tag register is used for any interrogating tools to determine if the device is being programmed or has been programmed. Read: Current value of claim tag. Write: 1= Clear claim tag bit, 0= No effect

Register ([funnel](#)) LAR

Name	LAR
Relative Address	0x0000FB0
Absolute Address	0xF8804FB0
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	Lock Access Register

Register LAR Details

Field Name	Bits	Type	Reset Value	Description
	31:0	wo	0x0	<p>Write Access Code.</p> <p>Write behavior depends on PADDRDBG31 pin:</p> <ul style="list-style-type: none"> - PADDRDBG31=0 (lower 2GB): <p>After reset (via PRESETDBGn), Funnel is locked, i.e., writes to all other registers using lower 2GB addresses are ignored.</p> <p>To unlock, 0xC5ACCE55 must be written this register.</p> <p>After the required registers are written, to lock again, write a value other than 0xC5ACCE55 to this register.</p> <ul style="list-style-type: none"> - PADDRDBG31=1 (upper 2GB): <p>Funnel is unlocked when upper 2GB addresses are used to write to all the registers.</p> <p>However, write to this register is ignored using a upper 2GB address!</p> <p>Note: read from this register always returns 0, regardless of PADDRDBG31.</p>

Register ([funnel](#)) LSR

Name	LSR
Relative Address	0x00000FB4
Absolute Address	0xF8804FB4
Width	3 bits
Access Type	ro
Reset Value	0x00000003
Description	Lock Status Register

Register LSR Details

Field Name	Bits	Type	Reset Value	Description
8BIT	2	ro	0x0	Set to 0 since Funnel implements a 32-bit lock access register
STATUS	1	ro	0x1	Read behavior depends on PADDRDBG31 pin: - PADDRDBG31=0 (lower 2GB): When a lower 2GB address is used to read this register, this bit indicates whether Funnel is in locked state (1= locked, 0= unlocked). - PADDRDBG31=1 (upper 2GB): always returns 0.
IMP	0	ro	0x1	Read behavior depends on PADDRDBG31 pin: - PADDRDBG31=0 (lower 2GB): always returns 1, meaning lock mechanism are implemented. - PADDRDBG31=1 (upper 2GB): always returns 0, meaning lock mechanism is NOT implemented.

Register ([funnel](#)) ASR

Name	ASR
Relative Address	0x00000FB8
Absolute Address	0xF8804FB8
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Authentication Status Register

Register ASR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	Indicates functionality not implemented

Register ([funnel](#)) DEVID

Name	DEVID
Relative Address	0x00000FC8
Absolute Address	0xF8804FC8
Width	8 bits

Access Type	ro
Reset Value	0x00000028
Description	Device ID

Register DEVID Details

Field Name	Bits	Type	Reset Value	Description
StaticPrio	7:4	ro	0x2	CSTF implements a static priority scheme
NumInPorts	3:0	ro	0x8	Number of input ports

Register ([funnel](#)) DTIR

Name	DTIR
Relative Address	0x00000FCC
Absolute Address	0xF8804FCC
Width	8 bits
Access Type	ro
Reset Value	0x00000012
Description	Device Type Identifier Register

Register DTIR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x12	a trace link and specifically a funnel/router

Register ([funnel](#)) PERIPHD4

Name	PERIPHD4
Relative Address	0x00000FD0
Absolute Address	0xF8804FD0
Width	8 bits
Access Type	ro
Reset Value	0x00000004
Description	Peripheral ID4

Register PERIPHD4 Details

Field Name	Bits	Type	Reset Value	Description
4KB_count	7:4	ro	0x0	4KB Count, set to 0
JEP106ID	3:0	ro	0x4	JEP106 continuation code

Register ([funnel](#)) PERIPHD5

Name	PERIPHD5
Relative Address	0x0000FD4
Absolute Address	0xF8804FD4
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID5

Register PERIPHD5 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([funnel](#)) PERIPHD6

Name	PERIPHD6
Relative Address	0x0000FD8
Absolute Address	0xF8804FD8
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID6

Register PERIPHD6 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([funnel](#)) PERIPHD7

Name	PERIPHD7
------	----------

Relative Address	0x00000FDC
Absolute Address	0xF8804FDC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID7

Register PERIPHD7 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([funnel](#)) PERIPHD0

Name	PERIPHD0
Relative Address	0x00000FE0
Absolute Address	0xF8804FE0
Width	8 bits
Access Type	ro
Reset Value	0x00000008
Description	Peripheral ID0

Register PERIPHD0 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x8	PartNumber0

Register ([funnel](#)) PERIPHD1

Name	PERIPHD1
Relative Address	0x00000FE4
Absolute Address	0xF8804FE4
Width	8 bits
Access Type	ro
Reset Value	0x000000B9
Description	Peripheral ID1

Register PERIPHD1 Details

Field Name	Bits	Type	Reset Value	Description
JEP106ID	7:4	ro	0xB	JEP106 Identity Code [3:0]
PartNumber1	3:0	ro	0x9	PartNumber1

Register ([funnel](#)) PERIPHD2

Name	PERIPHD2
Relative Address	0x0000FE8
Absolute Address	0xF8804FE8
Width	8 bits
Access Type	ro
Reset Value	0x000001B
Description	Peripheral ID2

Register PERIPHD2 Details

Field Name	Bits	Type	Reset Value	Description
RevNum	7:4	ro	0x1	Revision number of Peripheral
JEDEC	3	ro	0x1	Indicates that a JEDEC assigned value is used
JEP106ID	2:0	ro	0x3	JEP106 Identity Code [6:4]

Register ([funnel](#)) PERIPHD3

Name	PERIPHD3
Relative Address	0x0000FEC
Absolute Address	0xF8804FEC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID3

Register PERIPHD3 Details

Field Name	Bits	Type	Reset Value	Description
RevAnd	7:4	ro	0x0	RevAnd, at top level
CustMod	3:0	ro	0x0	Customer Modified

Register ([funnel](#)) COMPID0

Name	COMPID0
Relative Address	0x0000FF0
Absolute Address	0xF8804FF0
Width	8 bits
Access Type	ro
Reset Value	0x0000000D
Description	Component ID0

Register COMPID0 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0xD	Preamble

Register ([funnel](#)) COMPID1

Name	COMPID1
Relative Address	0x0000FF4
Absolute Address	0xF8804FF4
Width	8 bits
Access Type	ro
Reset Value	0x00000090
Description	Component ID1

Register COMPID1 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x90	Preamble

Register ([funnel](#)) COMPID2

Name	COMPID2
Relative Address	0x0000FF8
Absolute Address	0xF8804FF8
Width	8 bits
Access Type	ro
Reset Value	0x00000005
Description	Component ID2

Register COMPID2 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x5	Preamble

Register ([funnel](#)) COMPID3

Name	COMPID3
Relative Address	0x0000FFC
Absolute Address	0xF8804FFC
Width	8 bits
Access Type	ro
Reset Value	0x000000B1
Description	Component ID3

Register COMPID3 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0xB1	Preamble

B.14 CoreSight Intstrumentation Trace Macrocell (itm)

Module Name	CoreSight Intstrumentation Trace Macrocell (itm)
Base Address	0xF8805000 debug_itm
Description	Instrumentation Trace Macrocell
Version	0.65
Doc Version	
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
StimPort00	0x00000000	32	rw	0x00000000	Stimulus Port Register 0
StimPort01	0x00000004	32	rw	0x00000000	Stimulus Port Register 1
StimPort02	0x00000008	32	rw	0x00000000	Stimulus Port Register 2
StimPort03	0x0000000C	32	rw	0x00000000	Stimulus Port Register 3
StimPort04	0x00000010	32	rw	0x00000000	Stimulus Port Register 4
StimPort05	0x00000014	32	rw	0x00000000	Stimulus Port Register 5
StimPort06	0x00000018	32	rw	0x00000000	Stimulus Port Register 6
StimPort07	0x0000001C	32	rw	0x00000000	Stimulus Port Register 7
StimPort08	0x00000020	32	rw	0x00000000	Stimulus Port Register 8
StimPort09	0x00000024	32	rw	0x00000000	Stimulus Port Register 9
StimPort10	0x00000028	32	rw	0x00000000	Stimulus Port Register 10
StimPort11	0x0000002C	32	rw	0x00000000	Stimulus Port Register 11
StimPort12	0x00000030	32	rw	0x00000000	Stimulus Port Register 12
StimPort13	0x00000034	32	rw	0x00000000	Stimulus Port Register 13
StimPort14	0x00000038	32	rw	0x00000000	Stimulus Port Register 14
StimPort15	0x0000003C	32	rw	0x00000000	Stimulus Port Register 15
StimPort16	0x00000040	32	rw	0x00000000	Stimulus Port Register 16
StimPort17	0x00000044	32	rw	0x00000000	Stimulus Port Register 17
StimPort18	0x00000048	32	rw	0x00000000	Stimulus Port Register 18
StimPort19	0x0000004C	32	rw	0x00000000	Stimulus Port Register 19
StimPort20	0x00000050	32	rw	0x00000000	Stimulus Port Register 20

Register Name	Address	Width	Type	Reset Value	Description
StimPort21	0x00000054	32	rw	0x00000000	Stimulus Port Register 21
StimPort22	0x00000058	32	rw	0x00000000	Stimulus Port Register 22
StimPort23	0x0000005C	32	rw	0x00000000	Stimulus Port Register 23
StimPort24	0x00000060	32	rw	0x00000000	Stimulus Port Register 24
StimPort25	0x00000064	32	rw	0x00000000	Stimulus Port Register 25
StimPort26	0x00000068	32	rw	0x00000000	Stimulus Port Register 26
StimPort27	0x0000006C	32	rw	0x00000000	Stimulus Port Register 27
StimPort28	0x00000070	32	rw	0x00000000	Stimulus Port Register 28
StimPort29	0x00000074	32	rw	0x00000000	Stimulus Port Register 29
StimPort30	0x00000078	32	rw	0x00000000	Stimulus Port Register 30
StimPort31	0x0000007C	32	rw	0x00000000	Stimulus Port Register 31
TER	0x00000E00	32	rw	0x00000000	Trace Enable Register
TTR	0x00000E20	32	rw	0x00000000	Trace Trigger Register
CR	0x00000E80	24	mixed	0x00000004	Control Register
SCR	0x00000E90	12	rw	0x00000400	Synchronization Control Register
ITTRIGOUTACK	0x00000EE4	1	ro	0x00000000	Integration Test Trigger Out Acknowledge Register
ITTRIGOUT	0x00000EE8	1	wo	0x00000000	Integration Test Trigger Out Register
ITATBDATA0	0x00000EEC	2	wo	0x00000000	Integration Test ATB Data Register 0
ITATBCTR2	0x00000EF0	1	ro	0x00000001	Integration Test ATB Control Register 2
ITATABCTR1	0x00000EF4	7	wo	0x00000000	Integration Test ATB Control Register 1
ITATBCTR0	0x00000EF8	2	wo	0x00000000	Integration Test ATB Control Register 0
IMCR	0x00000F00	1	rw	0x00000000	Integration Mode Control Register
CTSR	0x00000FA0	8	rw	0x000000FF	Claim Tag Set Register
CTCR	0x00000FA4	8	rw	0x00000000	Claim Tag Clear Register
LAR	0x00000FB0	32	wo	0x00000000	Lock Access Register
LSR	0x00000FB4	3	ro	0x00000003	Lock Status Register
ASR	0x00000FB8	8	ro	0x00000088	Authentication Status Register
DEVID	0x00000FC8	13	ro	0x00000020	Device ID
DTIR	0x00000FCC	8	ro	0x00000043	Device Type Identifier Register

Register Name	Address	Width	Type	Reset Value	Description
PERIPHD4	0x00000FD0	8	ro	0x00000004	Peripheral ID4
PERIPHD5	0x00000FD4	8	ro	0x00000000	Peripheral ID5
PERIPHD6	0x00000FD8	8	ro	0x00000000	Peripheral ID6
PERIPHD7	0x00000FDC	8	ro	0x00000000	Peripheral ID7
PERIPHD0	0x00000FE0	8	ro	0x00000013	Peripheral ID0
PERIPHD1	0x00000FE4	8	ro	0x000000B9	Peripheral ID1
PERIPHD2	0x00000FE8	8	ro	0x0000002B	Peripheral ID2
PERIPHD3	0x00000FEC	8	ro	0x00000000	Peripheral ID3
COMPID0	0x00000FF0	8	ro	0x0000000D	Component ID0
COMPID1	0x00000FF4	8	ro	0x00000090	Component ID1
COMPID2	0x00000FF8	8	ro	0x00000005	Component ID2
COMPID3	0x00000FFC	8	ro	0x000000B1	Component ID3

Register ([itm](#)) StimPort00

Name	StimPort00
Relative Address	0x00000000
Absolute Address	0xF8805000
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 0

Register StimPort00 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort01

Name	StimPort01
Relative Address	0x00000004
Absolute Address	0xF8805004
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 1

Register StimPort01 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort02

Name	StimPort02
Relative Address	0x00000008
Absolute Address	0xF8805008
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 2

Register StimPort02 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort03

Name	StimPort03
Relative Address	0x0000000C
Absolute Address	0xF880500C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 3

Register StimPort03 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort04

Name	StimPort04
Relative Address	0x00000010
Absolute Address	0xF8805010
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 4

Register StimPort04 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort05

Name	StimPort05
Relative Address	0x00000014
Absolute Address	0xF8805014
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 5

Register StimPort05 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort06

Name	StimPort06
Relative Address	0x00000018
Absolute Address	0xF8805018
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 6

Register StimPort06 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort07

Name	StimPort07
Relative Address	0x0000001C
Absolute Address	0xF880501C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 7

Register StimPort07 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort08

Name	StimPort08
Relative Address	0x00000020
Absolute Address	0xF8805020
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 8

Register StimPort08 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort09

Name	StimPort09
Relative Address	0x00000024
Absolute Address	0xF8805024
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 9

Register StimPort09 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort10

Name	StimPort10
Relative Address	0x00000028
Absolute Address	0xF8805028
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 10

Register StimPort10 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort11

Name	StimPort11
Relative Address	0x0000002C
Absolute Address	0xF880502C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 11

Register StimPort11 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort12

Name	StimPort12
Relative Address	0x00000030
Absolute Address	0xF8805030
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 12

Register StimPort12 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort13

Name	StimPort13
Relative Address	0x00000034
Absolute Address	0xF8805034
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 13

Register StimPort13 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort14

Name	StimPort14
Relative Address	0x00000038
Absolute Address	0xF8805038
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 14

Register StimPort14 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort15

Name	StimPort15
Relative Address	0x0000003C
Absolute Address	0xF880503C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 15

Register StimPort15 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort16

Name	StimPort16
Relative Address	0x00000040
Absolute Address	0xF8805040
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 16

Register StimPort16 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort17

Name	StimPort17
Relative Address	0x00000044
Absolute Address	0xF8805044
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 17

Register StimPort17 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort18

Name	StimPort18
Relative Address	0x00000048
Absolute Address	0xF8805048
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 18

Register StimPort18 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort19

Name	StimPort19
Relative Address	0x0000004C
Absolute Address	0xF880504C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 19

Register StimPort19 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort20

Name	StimPort20
Relative Address	0x00000050
Absolute Address	0xF8805050
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 20

Register StimPort20 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort21

Name	StimPort21
Relative Address	0x00000054
Absolute Address	0xF8805054
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 21

Register StimPort21 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort22

Name	StimPort22
Relative Address	0x00000058
Absolute Address	0xF8805058
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 22

Register StimPort22 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort23

Name	StimPort23
Relative Address	0x0000005C
Absolute Address	0xF880505C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 23

Register StimPort23 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort24

Name	StimPort24
Relative Address	0x00000060
Absolute Address	0xF8805060
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 24

Register StimPort24 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort25

Name	StimPort25
Relative Address	0x00000064
Absolute Address	0xF8805064
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 25

Register StimPort25 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort26

Name	StimPort26
Relative Address	0x00000068
Absolute Address	0xF8805068
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 26

Register StimPort26 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort27

Name	StimPort27
Relative Address	0x0000006C
Absolute Address	0xF880506C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 27

Register StimPort27 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort28

Name	StimPort28
Relative Address	0x00000070
Absolute Address	0xF8805070
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 28

Register StimPort28 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort29

Name	StimPort29
Relative Address	0x00000074
Absolute Address	0xF8805074
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 29

Register StimPort29 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort30

Name	StimPort30
Relative Address	0x00000078
Absolute Address	0xF8805078
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 30

Register StimPort30 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) StimPort31

Name	StimPort31
Relative Address	0x0000007C
Absolute Address	0xF880507C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Stimulus Port Register 31

Register StimPort31 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Each of the 32 stimulus ports is represented by a virtual address, creating 32 stimulus registers. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set and ITM is enabled. Reading from any of the stimulus ports returns the FIFO status (notFull(1) / Full(0)) only if the ITM is enabled. This enables more efficient core register allocation because the stimulus address has already been generated.</p> <p>The ITM transmits SWIT packets using leading zero compression. Packets can be 8, 16, or 32 bits.</p> <p>The bank of 32 registers is split into a low-16 (0 to 15) and a high-16 (16 to 31). Writes to the high-16 are discarded by the ITM whenever secure non-invasive trace is disabled, regardless of how the Trace Enable Register bits [31:16] are set. Both the high-16 and</p> <p>low-16 are be disabled when non-invasive trace is disabled. When an input is disabled it must not alter the interface response and must always return an OK without stalling.</p>

Register ([itm](#)) TER

Name	TER
Relative Address	0x00000E00
Absolute Address	0xF8805E00
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Trace Enable Register

Register TER Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Bit mask to enable tracing on ITM stimulus ports.

Register ([itm](#)) TTR

Name	TTR
Relative Address	0x00000E20

Absolute Address	0xF8805E20
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Trace Trigger Register

Register TTR Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Bit mask to enable trigger generation, TRIGOUT, on selected writes to the Stimulus Registers.

Register ([itm](#)) CR

Name	CR
Relative Address	0x00000E80
Absolute Address	0xF8805E80
Width	24 bits
Access Type	mixed
Reset Value	0x00000004
Description	Control Register

Register CR Details

Field Name	Bits	Type	Reset Value	Description
ITMBusy	23	rw	0x0	ITM is transmitting trace and FIFO is not empty
TraceID	22:16	rw	0x0	ATIDM[6:0] value
reserved	15:10	ro	0x0	Reserved
TSPrescale	9:8	rw	0x0	Timestamp Prescaler Enumerated Value List: DIVBY1=0. DIVBY4=1. DIVBY16=2. DIVBY64=3.
reserved	7:4	ro	0x0	Reserved
DWTEn	3	ro	0x0	Enable DWT input port
SYNCEn	2	ro	0x1	Enable sync packets
TSSEn	1	rw	0x0	Enable timestamps, delta
ITMEn	0	rw	0x0	Enable ITM Stimulus, also acts as a global enable

Register ([itm](#)) SCR

Name	SCR
Relative Address	0x00000E90
Absolute Address	0xF8805E90
Width	12 bits
Access Type	rw
Reset Value	0x00000400
Description	Synchronization Control Register

Register SCR Details

Field Name	Bits	Type	Reset Value	Description
SyncCount	11:0	rw	0x400	Counter value for time between synchronization markers

Register ([itm](#)) ITTRIGOUTACK

Name	ITTRIGOUTACK
Relative Address	0x00000EE4
Absolute Address	0xF8805EE4
Width	1 bits
Access Type	ro
Reset Value	0x00000000
Description	Integration Test Trigger Out Acknowledge Register

Register ITTRIGOUTACK Details

Field Name	Bits	Type	Reset Value	Description
ITTRIGOUTACK	0	ro	0x0	Read the value of TRIGOUTACK

Register ([itm](#)) ITTRIGOUT

Name	ITTRIGOUT
Relative Address	0x00000EE8
Absolute Address	0xF8805EE8
Width	1 bits
Access Type	wo
Reset Value	0x00000000

Description Integration Test Trigger Out Register

Register ITTRIGOUT Details

Field Name	Bits	Type	Reset Value	Description
ITTRIGOUT	0	wo	0x0	Set the value of TRIGOUT

Register ([itm](#)) ITATBDATA0

Name ITATBDATA0

Relative Address 0x0000EEC

Absolute Address 0xF8805EEC

Width 2 bits

Access Type wo

Reset Value 0x00000000

Description Integration Test ATB Data Register 0

Register ITATBDATA0 Details

Field Name	Bits	Type	Reset Value	Description
ITATDATAM7	1	wo	0x0	Set the value of ATDATAM[7]
ITATDATAM0	0	wo	0x0	Set the value of ATDATAM[0]

Register ([itm](#)) ITATBCTR2

Name ITATBCTR2

Relative Address 0x0000EF0

Absolute Address 0xF8805EF0

Width 1 bits

Access Type ro

Reset Value 0x00000001

Description Integration Test ATB Control Register 2

Register ITATBCTR2 Details

Field Name	Bits	Type	Reset Value	Description
ITATREADYM	0	ro	0x1	Read the value of ATREADYM

Register ([itm](#)) ITATABCTR1

Name	ITATABCTR1
Relative Address	0x00000EF4
Absolute Address	0xF8805EF4
Width	7 bits
Access Type	wo
Reset Value	0x00000000
Description	Integration Test ATB Control Register 1

Register ITATABCTR1 Details

Field Name	Bits	Type	Reset Value	Description
ITATIDM	6:0	wo	0x0	Set the value of ATIDM[6:0]

Register ([itm](#)) ITATBCTR0

Name	ITATBCTR0
Relative Address	0x00000EF8
Absolute Address	0xF8805EF8
Width	2 bits
Access Type	wo
Reset Value	0x00000000
Description	Integration Test ATB Control Register 0

Register ITATBCTR0 Details

Field Name	Bits	Type	Reset Value	Description
ITAFREADYM	1	wo	0x0	Set the value of AFREADYM
ITATVALIDM	0	wo	0x0	Set the value of ATVALIDM

Register ([itm](#)) IMCR

Name	IMCR
Relative Address	0x00000F00
Absolute Address	0xF8805F00
Width	1 bits
Access Type	rw
Reset Value	0x00000000

Description Integration Mode Control Register

Register IMCR Details

Field Name	Bits	Type	Reset Value	Description
	0	rw	0x0	Enable Integration Test registers.

Register ([itm](#)) CTSR

Name CTSR

Relative Address 0x00000FA0

Absolute Address 0xF8805FA0

Width 8 bits

Access Type rw

Reset Value 0x000000FF

Description Claim Tag Set Register

Register CTSR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	rw	0xFF	The claim tag register is used for any interrogating tools to determine if the device is being programmed or has been programmed. Read: 1= Claim tag is implemented, 0 = Claim tag is not implemented Write: 1= Set claim tag bit, 0= No effect

Register ([itm](#)) CTCR

Name CTCR

Relative Address 0x00000FA4

Absolute Address 0xF8805FA4

Width 8 bits

Access Type rw

Reset Value 0x00000000

Description Claim Tag Clear Register

Register CTCR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	rw	0x0	The claim tag register is used for any interrogating tools to determine if the device is being programmed or has been programmed. Read: Current value of claim tag. Write: 1= Clear claim tag bit, 0= No effect

Register ([itm](#)) LAR

Name	LAR
Relative Address	0x00000FB0
Absolute Address	0xF8805FB0
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	Lock Access Register

Register LAR Details

Field Name	Bits	Type	Reset Value	Description
	31:0	wo	0x0	Write Access Code. Write behavior depends on PADDRDBG31 pin: - PADDRDBG31=0 (lower 2GB): After reset (via PRESETDBGn), ITM is locked, i.e., writes to all other registers using lower 2GB addresses are ignored. To unlock, 0xC5ACCE55 must be written this register. After the required registers are written, to lock again, write a value other than 0xC5ACCE55 to this register. - PADDRDBG31=1 (upper 2GB): ITM is unlocked when upper 2GB addresses are used to write to all the registers. However, write to this register is ignored using a upper 2GB address! Note: read from this register always returns 0, regardless of PADDRDBG31.

Register ([itm](#)) LSR

Name	LSR
------	-----

Relative Address	0x00000FB4
Absolute Address	0xF8805FB4
Width	3 bits
Access Type	ro
Reset Value	0x00000003
Description	Lock Status Register

Register LSR Details

Field Name	Bits	Type	Reset Value	Description
8BIT	2	ro	0x0	Set to 0 since ITM implements a 32-bit lock access register
STATUS	1	ro	0x1	Read behavior depends on PADDRDBG31 pin: - PADDRDBG31=0 (lower 2GB): When a lower 2GB address is used to read this register, this bit indicates whether ITM is in locked state (1= locked, 0= unlocked). - PADDRDBG31=1 (upper 2GB): always returns 0.
IMP	0	ro	0x1	Read behavior depends on PADDRDBG31 pin: - PADDRDBG31=0 (lower 2GB): always returns 1, meaning lock mechanism are implemented. - PADDRDBG31=1 (upper 2GB): always returns 0, meaning lock mechanism is NOT implemented.

Register ([itm](#)) ASR

Name	ASR
Relative Address	0x00000FB8
Absolute Address	0xF8805FB8
Width	8 bits
Access Type	ro
Reset Value	0x00000088
Description	Authentication Status Register

Register ASR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x88	Value is 0b1S001N00 where S is secure non-invasive debug state and N is non-secure, non-invasive debug.

Register ([itm](#)) DEVID

Name	DEVID
Relative Address	0x00000FC8
Absolute Address	0xF8805FC8
Width	13 bits
Access Type	ro
Reset Value	0x00000020
Description	Device ID

Register DEVID Details

Field Name	Bits	Type	Reset Value	Description
NumStimRegs	12:0	ro	0x20	Number of stimulus registers

Register ([itm](#)) DTIR

Name	DTIR
Relative Address	0x00000FCC
Absolute Address	0xF8805FCC
Width	8 bits
Access Type	ro
Reset Value	0x00000043
Description	Device Type Identifier Register

Register DTIR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x43	Indicates a Trace Source and the stimulus is derived from bus activity

Register ([itm](#)) PERIPID4

Name	PERIPID4
------	----------

Relative Address	0x0000FD0
Absolute Address	0xF8805FD0
Width	8 bits
Access Type	ro
Reset Value	0x00000004
Description	Peripheral ID4

Register PERIPHD4 Details

Field Name	Bits	Type	Reset Value	Description
4KB_count	7:4	ro	0x0	4KB Count, set to 0
JEP106ID	3:0	ro	0x4	JEP106 continuation code

Register (itm) PERIPHD5

Name	PERIPHD5
Relative Address	0x0000FD4
Absolute Address	0xF8805FD4
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID5

Register PERIPHD5 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register (itm) PERIPHD6

Name	PERIPHD6
Relative Address	0x0000FD8
Absolute Address	0xF8805FD8
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID6

Register PERIPHD6 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([itm](#)) PERIPHD7

Name	PERIPHD7
Relative Address	0x0000FDC
Absolute Address	0xF8805FDC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID7

Register PERIPHD7 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([itm](#)) PERIPHD0

Name	PERIPHD0
Relative Address	0x0000FE0
Absolute Address	0xF8805FE0
Width	8 bits
Access Type	ro
Reset Value	0x00000013
Description	Peripheral ID0

Register PERIPHD0 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x13	PartNumber0

Register ([itm](#)) PERIPHD1

Name	PERIPHD1
Relative Address	0x0000FE4

Absolute Address 0xF8805FE4
 Width 8 bits
 Access Type ro
 Reset Value 0x000000B9
 Description Peripheral ID1

Register PERIPHD1 Details

Field Name	Bits	Type	Reset Value	Description
JEP106ID	7:4	ro	0xB	JEP106 Identity Code [3:0]
PartNumber1	3:0	ro	0x9	PartNumber1

Register (itm) PERIPHD2

Name PERIPHD2
 Relative Address 0x00000FE8
 Absolute Address 0xF8805FE8
 Width 8 bits
 Access Type ro
 Reset Value 0x0000002B
 Description Peripheral ID2

Register PERIPHD2 Details

Field Name	Bits	Type	Reset Value	Description
RevNum	7:4	ro	0x2	Revision number of Peripheral
JEDEC	3	ro	0x1	Indicates that a JEDEC assigned value is used
JEP106ID	2:0	ro	0x3	JEP106 Identity Code [6:4]

Register (itm) PERIPHD3

Name PERIPHD3
 Relative Address 0x00000FEC
 Absolute Address 0xF8805FEC
 Width 8 bits
 Access Type ro
 Reset Value 0x00000000
 Description Peripheral ID3

Register PERIPID3 Details

Field Name	Bits	Type	Reset Value	Description
RevAnd	7:4	ro	0x0	RevAnd, at top level
CustMod	3:0	ro	0x0	Customer Modified

Register ([itm](#)) COMPID0

Name	COMPID0
Relative Address	0x00000FF0
Absolute Address	0xF8805FF0
Width	8 bits
Access Type	ro
Reset Value	0x0000000D
Description	Component ID0

Register COMPID0 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0xD	Preamble

Register ([itm](#)) COMPID1

Name	COMPID1
Relative Address	0x00000FF4
Absolute Address	0xF8805FF4
Width	8 bits
Access Type	ro
Reset Value	0x00000090
Description	Component ID1

Register COMPID1 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x90	Preamble

Register ([itm](#)) COMPID2

Name	COMPID2
------	---------

Relative Address	0x0000FF8
Absolute Address	0xF8805FF8
Width	8 bits
Access Type	ro
Reset Value	0x00000005
Description	Component ID2

Register COMPID2 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x5	Preamble

Register ([itm](#)) COMPID3

Name	COMPID3
Relative Address	0x0000FFC
Absolute Address	0xF8805FFC
Width	8 bits
Access Type	ro
Reset Value	0x000000B1
Description	Component ID3

Register COMPID3 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0xB1	Preamble

B.15 CoreSight Trace Packet Output (tpiu)

Module Name	CoreSight Trace Packet Output (tpiu)
Base Address	0xF8803000 debug_tpiu
Description	Trace Port Interface Unit
Version	0.65
Doc Version	
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
SuppSize	0x00000000	32	rw	0xFFFFFFFF	Supported Port Size Register
CurrentSize	0x00000004	32	rw	0x00000001	Current Port Size Register
SuppTrigMode	0x00000100	18	ro	0x0000011F	Supported Trigger Modes Register
TrigCount	0x00000104	8	rw	0x00000000	Trigger Counter Register
TrigMult	0x00000108	5	rw	0x00000000	Trigger Multiplier Register
SuppTest	0x00000200	18	ro	0x0003000F	Supported Test Patterns/Modes Register
CurrentTest	0x00000204	18	mixed	0x00000000	Current Test Patterns/Modes Register Only one of the modes can be set using bits 17-16, but a multiple number of bits for the patterns can be set using bits 3-0.
TestRepeatCount	0x00000208	8	rw	0x00000000	TPIU Test Pattern Repeat Counter Register
FFSR	0x00000300	3	ro	0x00000006	Formatter and Flush Status Register
FFCR	0x00000304	14	mixed	0x00000000	Formatter and Flush Control Register
FormatSyncCount	0x00000308	12	rw	0x00000040	Formatter Synchronization Counter Register
EXTCTLIn	0x00000400	8	ro	0x00000000	EXTCTL In Port
EXTCTLOut	0x00000404	8	rw	0x00000000	EXTCTL Out Port
ITTRFLINACK	0x00000EE4	2	wo	0x00000000	Integration Test Trigger In and Flush In Acknowledge Register

Register Name	Address	Width	Type	Reset Value	Description
ITTRFLIN	0x00000EE8	2	ro	x	Integration Test Trigger In and Flush In Register
ITATBDATA0	0x00000EEC	5	ro	x	Integration Test ATB Data Register 0
ITATBCTR2	0x00000EF0	2	wo	0x00000000	Integration Test ATB Control Register 2
ITATBCTR1	0x00000EF4	7	ro	x	Integration Test ATB Control Register 1
ITATBCTR0	0x00000EF8	10	ro	x	Integration Test ATB Control Register 0
IMCR	0x00000F00	1	rw	0x00000000	Integration Mode Control Register
CTSR	0x00000FA0	4	rw	0x0000000F	Claim Tag Set Register
CTCR	0x00000FA4	4	rw	0x00000000	Claim Tag Clear Register
LAR	0x00000FB0	32	wo	0x00000000	Lock Access Register
LSR	0x00000FB4	3	ro	0x00000003	Lock Status Register
ASR	0x00000FB8	8	ro	0x00000000	Authentication Status Register
DEVID	0x00000FC8	12	ro	0x000000A0	Device ID
DTIR	0x00000FCC	8	ro	0x00000011	Device Type Identifier Register
PERIPHID4	0x00000FD0	8	ro	0x00000004	Peripheral ID4
PERIPHID5	0x00000FD4	8	ro	0x00000000	Peripheral ID5
PERIPHID6	0x00000FD8	8	ro	0x00000000	Peripheral ID6
PERIPHID7	0x00000FDC	8	ro	0x00000000	Peripheral ID7
PERIPHID0	0x00000FE0	8	ro	0x00000012	Peripheral ID0
PERIPHID1	0x00000FE4	8	ro	0x000000B9	Peripheral ID1
PERIPHID2	0x00000FE8	8	ro	0x0000004B	Peripheral ID2
PERIPHID3	0x00000FEC	8	ro	0x00000000	Peripheral ID3
COMPID0	0x00000FF0	8	ro	0x0000000D	Component ID0
COMPID1	0x00000FF4	8	ro	0x00000090	Component ID1
COMPID2	0x00000FF8	8	ro	0x00000005	Component ID2
COMPID3	0x00000FFC	8	ro	0x000000B1	Component ID3

Register ([tpiu](#)) SuppSize

Name SuppSize
Relative Address 0x00000000

Absolute Address	0xF8803000
Width	32 bits
Access Type	rw
Reset Value	0xFFFFFFFF
Description	Supported Port Size Register

Register SuppSize Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0xFFFFFFFF	Each bit location represents a single port size that is supported on the device, that is, 32-1 in bit locations [31:0].

Register ([tpiu](#)) CurrentSize

Name	CurrentSize
Relative Address	0x00000004
Absolute Address	0xF8803004
Width	32 bits
Access Type	rw
Reset Value	0x00000001
Description	Current Port Size Register

Register CurrentSize Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x1	The Current Port Size Register has the same format as the Supported Port Sizes register but only one bit is set, and all others must be zero. Writing values with more than one bit set or setting a bit that is not indicated as supported is not supported and causes unpredictable behavior.

Register ([tpiu](#)) SuppTrigMode

Name	SuppTrigMode
Relative Address	0x00000100
Absolute Address	0xF8803100
Width	18 bits
Access Type	ro

Reset Value 0x0000011F

Description Supported Trigger Modes Register

Register SuppTrigMode Details

Field Name	Bits	Type	Reset Value	Description
TrgRun	17	ro	0x0	Trigger Counter running. A trigger has occurred but the counter is not at zero.
Triggered	16	ro	0x0	A trigger has occurred and the counter has reached zero.
reserved	15:9	ro	0x0	Reserved
TCount8	8	ro	0x1	8-bit wide counter register implemented.
reserved	7:5	ro	0x0	Reserved
Mult64k	4	ro	0x1	Multiply the Trigger Counter by 65536 supported.
Mult256	3	ro	0x1	Multiply the Trigger Counter by 256 supported.
Mult16	2	ro	0x1	Multiply the Trigger Counter by 16 supported.
Mult4	1	ro	0x1	Multiply the Trigger Counter by 4 supported.
Mult2	0	ro	0x1	Multiply the Trigger Counter by 2 supported.

Register ([tpiu](#)) TrigCount

Name TrigCount

Relative Address 0x00000104

Absolute Address 0xF8803104

Width 8 bits

Access Type rw

Reset Value 0x00000000

Description Trigger Counter Register

Register TrigCount Details

Field Name	Bits	Type	Reset Value	Description
TrigCount	7:0	rw	0x0	8-bit counter value for the number of words to be output from the formatter before a trigger is inserted.

Register ([tpiu](#)) TrigMult

Name TrigMult

Relative Address 0x00000108

Absolute Address	0xF8803108
Width	5 bits
Access Type	rw
Reset Value	0x00000000
Description	Trigger Multiplier Register

Register TrigMult Details

Field Name	Bits	Type	Reset Value	Description
Mult64k	4	rw	0x0	Multiply the Trigger Counter by 65536.
Mult256	3	rw	0x0	Multiply the Trigger Counter by 256.
Mult16	2	rw	0x0	Multiply the Trigger Counter by 16.
Mult4	1	rw	0x0	Multiply the Trigger Counter by 4.
Mult2	0	rw	0x0	Multiply the Trigger Counter by 2.

Register ([tpiu](#)) SuppTest

Name	SuppTest
Relative Address	0x00000200
Absolute Address	0xF8803200
Width	18 bits
Access Type	ro
Reset Value	0x0003000F
Description	Supported Test Patterns/Modes Register

Register SuppTest Details

Field Name	Bits	Type	Reset Value	Description
PContEn	17	ro	0x1	Continuous mode.
PTimeEn	16	ro	0x1	Timed mode.
reserved	15:4	ro	0x0	Reserved
PatF0	3	ro	0x1	FF/00 Pattern
PatA5	2	ro	0x1	AA/55 Pattern
PatW0	1	ro	0x1	Walking 0s Pattern
PatW1	0	ro	0x1	Walking 1s Pattern

Register ([tpiu](#)) CurrentTest

Name	CurrentTest
Relative Address	0x00000204
Absolute Address	0xF8803204
Width	18 bits
Access Type	mixed
Reset Value	0x00000000
Description	Current Test Patterns/Modes Register Only one of the modes can be set using bits 17-16, but a multiple number of bits for the patterns can be set using bits 3-0.

Register CurrentTest Details

Field Name	Bits	Type	Reset Value	Description
PContEn	17	rw	0x0	Continuous mode.
PTimeEn	16	rw	0x0	Timed mode.
reserved	15:4	ro	0x0	Reserved
PatF0	3	rw	0x0	FF/00 Pattern
PatA5	2	rw	0x0	AA/55 Pattern
PatW0	1	rw	0x0	Walking 0s Pattern
PatW1	0	rw	0x0	Walking 1s Pattern

Register ([tpiu](#)) TestRepeatCount

Name	TestRepeatCount
Relative Address	0x00000208
Absolute Address	0xF8803208
Width	8 bits
Access Type	rw
Reset Value	0x00000000
Description	TPIU Test Pattern Repeat Counter Register

Register TestRepeatCount Details

Field Name	Bits	Type	Reset Value	Description
PattCount	7:0	rw	0x0	8-bit counter value to indicate the number of TRACECLKIN cycles that a pattern runs for before switching to the next pattern.

Register ([tpiu](#)) FFSR

Name	FFSR
Relative Address	0x00000300
Absolute Address	0xF8803300
Width	3 bits
Access Type	ro
Reset Value	0x00000006
Description	Formatter and Flush Status Register

Register FFSR Details

Field Name	Bits	Type	Reset Value	Description
TCPresent	2	ro	0x1	If this bit is set then TRACECTL is present.
FtStopped	1	ro	0x1	Formatter stopped. The formatter has received a stop request signal and all trace data and post-amble has been output. Any more trace data on the ATB interface is ignored and ATREADY goes HIGH.
FlInProg	0	ro	0x0	Flush In Progress. This is an indication of the current state of AFVALIDS.

Register ([tpiu](#)) FFCR

Name	FFCR
Relative Address	0x00000304
Absolute Address	0xF8803304
Width	14 bits
Access Type	mixed
Reset Value	0x00000000
Description	Formatter and Flush Control Register

Register FFCR Details

Field Name	Bits	Type	Reset Value	Description
StopTrig	13	rw	0x0	Stop the formatter after a Trigger Event is observed.
StopFl	12	rw	0x0	Stop the formatter after a flush completes (return of AFREADY). This forces the FIFO to drain off any part-completed packets.
reserved	11	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
TrigFl	10	rw	0x0	Indicates a trigger on Flush completion on AFREADY being returned.
TrigEvt	9	rw	0x0	Indicates a trigger on a Trigger Event.
TrigIn	8	rw	0x0	Indicates a trigger on TRIGIN being asserted.
reserved	7	ro	0x0	Reserved
FOnMan	6	rw	0x0	Manually generate a flush of the system. Setting this bit causes a flush to be generated. This is cleared when this flush has been serviced.
FOnTrig	5	rw	0x0	Generate a flush using Trigger event. Set this bit to cause a flush of data in the system when a Trigger Event occurs.
FOnFlIn	4	rw	0x0	Generate flush using the FLUSHIN interface. Set this bit to enable use of the FLUSHIN connection.
reserved	3:2	ro	0x0	Reserved
EnFCont	1	rw	0x0	Continuous Formatting, no TRACECTL. Embed in trigger packets and indicate null cycles using Sync packets. Can only be changed when FtStopped is HIGH.
EnFTC	0	rw	0x0	Enable Formatting. Do not embed Triggers into the formatted stream. Trace disable cycles and triggers are indicated by TRACECTL, where fitted. Can only be changed when FtStopped is HIGH.

Register ([tpiu](#)) FormatSyncCount

Name	FormatSyncCount
Relative Address	0x00000308
Absolute Address	0xF8803308
Width	12 bits
Access Type	rw
Reset Value	0x00000040
Description	Formatter Synchronization Counter Register

Register FormatSyncCount Details

Field Name	Bits	Type	Reset Value	Description
CycCount	11:0	rw	0x40	12-bit counter value to indicate the number of complete frames between full synchronization packets.

Register ([tpiu](#)) EXTCTLIn

Name	EXTCTLIn
Relative Address	0x00000400
Absolute Address	0xF8803400
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	EXTCTL In Port

Register EXTCTLIn Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	Tied to 0

Register ([tpiu](#)) EXTCTLOut

Name	EXTCTLOut
Relative Address	0x00000404
Absolute Address	0xF8803404
Width	8 bits
Access Type	rw
Reset Value	0x00000000
Description	EXTCTL Out Port

Register EXTCTLOut Details

Field Name	Bits	Type	Reset Value	Description
	7:0	rw	0x0	Output not connected

Register ([tpiu](#)) ITTRFLINACK

Name	ITTRFLINACK
Relative Address	0x00000EE4
Absolute Address	0xF8803EE4
Width	2 bits
Access Type	wo
Reset Value	0x00000000
Description	Integration Test Trigger In and Flush In Acknowledge Register

Register ITTRFLINACK Details

Field Name	Bits	Type	Reset Value	Description
FLUSHINACK	1	wo	0x0	Set the value of FLUSHINACK
TRIGINACK	0	wo	0x0	Set the value of TRIGINACK

Register ([tpiu](#)) ITTRFLIN

Name	ITTRFLIN
Relative Address	0x00000EE8
Absolute Address	0xF8803EE8
Width	2 bits
Access Type	ro
Reset Value	x
Description	Integration Test Trigger In and Flush In Register

Register ITTRFLIN Details

Field Name	Bits	Type	Reset Value	Description
FLUSHIN	1	ro	x	Read the value of FLUSHIN
TRIGIN	0	ro	x	Read the value of TRIGIN

Register ([tpiu](#)) ITATBDATA0

Name	ITATBDATA0
Relative Address	0x00000EEC
Absolute Address	0xF8803EEC
Width	5 bits
Access Type	ro
Reset Value	x
Description	Integration Test ATB Data Register 0

Register ITATBDATA0 Details

Field Name	Bits	Type	Reset Value	Description
ATDATA31	4	ro	x	Read the value of ATDATAS[31]
ATDATA23	3	ro	x	Read the value of ATDATAS[23]
ATDATA15	2	ro	x	Read the value of ATDATAS[15]

Field Name	Bits	Type	Reset Value	Description
ATDATA7	1	ro	x	Read the value of ATDATAS[7]
ATDATA0	0	ro	x	Read the value of ATDATAS[0]

Register ([tpiu](#)) ITATBCTR2

Name	ITATBCTR2
Relative Address	0x00000EF0
Absolute Address	0xF8803EF0
Width	2 bits
Access Type	wo
Reset Value	0x00000000
Description	Integration Test ATB Control Register 2

Register ITATBCTR2 Details

Field Name	Bits	Type	Reset Value	Description
AFVALID	1	wo	0x0	Set the value of AFVALIDS
ATREADY	0	wo	0x0	Set the value of ATREADYD

Register ([tpiu](#)) ITATBCTR1

Name	ITATBCTR1
Relative Address	0x00000EF4
Absolute Address	0xF8803EF4
Width	7 bits
Access Type	ro
Reset Value	x
Description	Integration Test ATB Control Register 1

Register ITATBCTR1 Details

Field Name	Bits	Type	Reset Value	Description
ATID	6:0	ro	x	Read the value of ATIDS

Register ([tpiu](#)) ITATBCTR0

Name	ITATBCTR0
------	-----------

Relative Address	0x00000EF8
Absolute Address	0xF8803EF8
Width	10 bits
Access Type	ro
Reset Value	x
Description	Integration Test ATB Control Register 0

Register ITATBCTR0 Details

Field Name	Bits	Type	Reset Value	Description
ATBYTES	9:8	ro	x	Read the value of ATBYTESS
reserved	7:2	ro	x	Reserved
AFREADY	1	ro	x	Read the value of AFREADY
ATVALID	0	ro	x	Read the value of ATVALID

Register (tpiu) IMCR

Name	IMCR
Relative Address	0x00000F00
Absolute Address	0xF8803F00
Width	1 bits
Access Type	rw
Reset Value	0x00000000
Description	Integration Mode Control Register

Register IMCR Details

Field Name	Bits	Type	Reset Value	Description
	0	rw	0x0	Enable Integration Test registers

Register (tpiu) CTSR

Name	CTSR
Relative Address	0x00000FA0
Absolute Address	0xF8803FA0
Width	4 bits
Access Type	rw
Reset Value	0x0000000F

Description Claim Tag Set Register

Register CTSR Details

Field Name	Bits	Type	Reset Value	Description
	3:0	rw	0xF	The claim tag register is used for any interrogating tools to determine if the device is being programmed or has been programmed. Read: 1= Claim tag is implemented, 0 = Claim tag is not implemented Write: 1= Set claim tag bit, 0= No effect

Register ([tpiu](#)) CTCR

Name CTCR
 Relative Address 0x00000FA4
 Absolute Address 0xF8803FA4
 Width 4 bits
 Access Type rw
 Reset Value 0x00000000
 Description Claim Tag Clear Register

Register CTCR Details

Field Name	Bits	Type	Reset Value	Description
	3:0	rw	0x0	The claim tag register is used for any interrogating tools to determine if the device is being programmed or has been programmed. Read: Current value of claim tag. Write: 1= Clear claim tag bit, 0= No effect

Register ([tpiu](#)) LAR

Name LAR
 Relative Address 0x00000FB0
 Absolute Address 0xF8803FB0
 Width 32 bits
 Access Type wo
 Reset Value 0x00000000

Description Lock Access Register

Register LAR Details

Field Name	Bits	Type	Reset Value	Description
	31:0	wo	0x0	<p>Write Access Code.</p> <p>Write behavior depends on PADDRDBG31 pin:</p> <ul style="list-style-type: none"> - PADDRDBG31=0 (lower 2GB): <p>After reset (via PRESETDBGn), TPIU is locked, i.e., writes to all other registers using lower 2GB addresses are ignored.</p> <p>To unlock, 0xC5ACCE55 must be written this register.</p> <p>After the required registers are written, to lock again, write a value other than 0xC5ACCE55 to this register.</p> <ul style="list-style-type: none"> - PADDRDBG31=1 (upper 2GB): <p>TPIU is unlocked when upper 2GB addresses are used to write to all the registers.</p> <p>However, write to this register is ignored using a upper 2GB address!</p> <p>Note: read from this register always returns 0, regardless of PADDRDBG31.</p>

Register ([tpiu](#)) LSR

Name	LSR
Relative Address	0x00000FB4
Absolute Address	0xF8803FB4
Width	3 bits
Access Type	ro
Reset Value	0x00000003
Description	Lock Status Register

Register LSR Details

Field Name	Bits	Type	Reset Value	Description
8BIT	2	ro	0x0	Set to 0 since TPIU implements a 32-bit lock access register
STATUS	1	ro	0x1	Read behavior depends on PADDRDBG31 pin: - PADDRDBG31=0 (lower 2GB): When a lower 2GB address is used to read this register, this bit indicates whether TPIU is in locked state (1= locked, 0= unlocked). - PADDRDBG31=1 (upper 2GB): always returns 0.
IMP	0	ro	0x1	Read behavior depends on PADDRDBG31 pin: - PADDRDBG31=0 (lower 2GB): always returns 1, meaning lock mechanism are implemented. - PADDRDBG31=1 (upper 2GB): always returns 0, meaning lock mechanism is NOT implemented.

Register ([tpiu](#)) ASR

Name	ASR
Relative Address	0x00000FB8
Absolute Address	0xF8803FB8
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Authentication Status Register

Register ASR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	Indicates functionality not implemented

Register ([tpiu](#)) DEVID

Name	DEVID
Relative Address	0x00000FC8
Absolute Address	0xF8803FC8
Width	12 bits

Access Type	ro
Reset Value	0x000000A0
Description	Device ID

Register DEVID Details

Field Name	Bits	Type	Reset Value	Description
UartNRZ	11	ro	0x0	UART/NRZ not supported
Manchester	10	ro	0x0	Manchester not support
ClockData	9	ro	0x0	Trace clock + data is supported
FifoSize	8:6	ro	0x2	FIFO size is 4
AsyncClock	5	ro	0x1	ATCLK and TRACECLKIN is asynchronous
InputMux	4:0	ro	0x0	No input multiplexing

Register ([tpiu](#)) DTIR

Name	DTIR
Relative Address	0x00000FCC
Absolute Address	0xF8803FCC
Width	8 bits
Access Type	ro
Reset Value	0x00000011
Description	Device Type Identifier Register

Register DTIR Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x11	A trace sink and specifically a TPIU

Register ([tpiu](#)) PERIPID4

Name	PERIPID4
Relative Address	0x00000FD0
Absolute Address	0xF8803FD0
Width	8 bits
Access Type	ro
Reset Value	0x00000004
Description	Peripheral ID4

Register PERIPHD4 Details

Field Name	Bits	Type	Reset Value	Description
4KB_count	7:4	ro	0x0	4KB Count, set to 0
JEP106ID	3:0	ro	0x4	JEP106 continuation code

Register ([tpiu](#)) PERIPHD5

Name	PERIPHD5
Relative Address	0x0000FD4
Absolute Address	0xF8803FD4
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID5

Register PERIPHD5 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([tpiu](#)) PERIPHD6

Name	PERIPHD6
Relative Address	0x0000FD8
Absolute Address	0xF8803FD8
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID6

Register PERIPHD6 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([tpiu](#)) PERIPHD7

Name	PERIPHD7
------	----------

Relative Address	0x00000FDC
Absolute Address	0xF8803FDC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID7

Register PERIPHD7 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x0	reserved

Register ([tpiu](#)) PERIPHD0

Name	PERIPHD0
Relative Address	0x00000FE0
Absolute Address	0xF8803FE0
Width	8 bits
Access Type	ro
Reset Value	0x00000012
Description	Peripheral ID0

Register PERIPHD0 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x12	PartNumber0

Register ([tpiu](#)) PERIPHD1

Name	PERIPHD1
Relative Address	0x00000FE4
Absolute Address	0xF8803FE4
Width	8 bits
Access Type	ro
Reset Value	0x000000B9
Description	Peripheral ID1

Register PERIPHD1 Details

Field Name	Bits	Type	Reset Value	Description
JEP106ID	7:4	ro	0xB	JEP106 Identity Code [3:0]
PartNumber1	3:0	ro	0x9	PartNumber1

Register ([tpiu](#)) PERIPHD2

Name	PERIPHD2
Relative Address	0x0000FE8
Absolute Address	0xF8803FE8
Width	8 bits
Access Type	ro
Reset Value	0x0000004B
Description	Peripheral ID2

Register PERIPHD2 Details

Field Name	Bits	Type	Reset Value	Description
RevNum	7:4	ro	0x4	Revision number of Peripheral
JEDEC	3	ro	0x1	Indicates that a JEDEC assigned value is used
JEP106ID	2:0	ro	0x3	JEP106 Identity Code [6:4]

Register ([tpiu](#)) PERIPHD3

Name	PERIPHD3
Relative Address	0x0000FEC
Absolute Address	0xF8803FEC
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Peripheral ID3

Register PERIPHD3 Details

Field Name	Bits	Type	Reset Value	Description
RevAnd	7:4	ro	0x0	RevAnd, at top level
CustMod	3:0	ro	0x0	Customer Modified

Register ([tpiu](#)) COMPID0

Name	COMPID0
Relative Address	0x0000FF0
Absolute Address	0xF8803FF0
Width	8 bits
Access Type	ro
Reset Value	0x0000000D
Description	Component ID0

Register COMPID0 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0xD	Preamble

Register ([tpiu](#)) COMPID1

Name	COMPID1
Relative Address	0x0000FF4
Absolute Address	0xF8803FF4
Width	8 bits
Access Type	ro
Reset Value	0x00000090
Description	Component ID1

Register COMPID1 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x90	Preamble

Register ([tpiu](#)) COMPID2

Name	COMPID2
Relative Address	0x0000FF8
Absolute Address	0xF8803FF8
Width	8 bits
Access Type	ro
Reset Value	0x00000005
Description	Component ID2

Register COMPID2 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0x5	Preamble

Register ([tpiu](#)) COMPID3

Name	COMPID3
Relative Address	0x0000FFC
Absolute Address	0xF8803FFC
Width	8 bits
Access Type	ro
Reset Value	0x000000B1
Description	Component ID3

Register COMPID3 Details

Field Name	Bits	Type	Reset Value	Description
	7:0	ro	0xB1	Preamble

B.16 Device Configuration Interface (devcfg)

Module Name	Device Configuration Interface (devcfg)
Software Name	XDCFG
Base Address	0xF8007000 devcfg
Description	Device configuraion Interface
Version	1.3
Doc Version	1.1
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
CTRL	0x00000000	32	mixed	0x0C000000	Control Register : This register defines basic control registers. Some of the register bits can be locked by control bits in the LOCK Register 0x004.
LOCK	0x00000004	32	mixed	0x00000000	This register defines LOCK register used to lock changes in the Control Register 0x000 after configuration. All those LOCK register is set only register. The only way to clear those registers is power on reset signal.
CFG	0x00000008	32	rw	0x0000050B	Configuration Register : This register contains configuration information for the AXI transfers, and other general setup.

Register Name	Address	Width	Type	Reset Value	Description
INT_STS	0x0000000C	32	mixed	0x00000000	<p>Interrupt Status Register : This register contains interrupt status flags.</p> <p>All register bits are clear on write by writing 1s to those bits, however the register bits will only be cleared if the condition that sets the interrupt flag is no longer true.</p> <p>Note that individual status bits will be set if the corresponding condition is satisfied regardless of whether the interrupt mask bit in 0x010 is set.</p> <p>However, external interrupt will only be generated if an interrupt status flag is set and the corresponding mask bit is not set</p>
INT_MASK	0x00000010	32	rw	0xFFFFFFFF	<p>Interrupt Mask Register: This register contains interrupt mask information.</p> <p>Set a bit to 1 to mask the interrupt generation from the corresponding interrupting source in Interrupt Status Register 0x00C.</p>
STATUS	0x00000014	32	mixed	0x40000820	Status Register: This register contains miscellaneous status.
DMA_SRC_ADDR	0x00000018	32	rw	0x00000000	<p>DMA Source address Register: This register contains the source address for DMA transfer.</p> <p>A DMA command consists of source address, destination address, source transfer length, and destination transfer length.</p> <p>It is important that the parameters are programmed in the exact sequence as described</p>

Register Name	Address	Width	Type	Reset Value	Description
DMA_DST_ADDR	0x0000001C	32	rw	0x00000000	<p>DMA Destination address Register: This register contains the destination address for DMA transfer.</p> <p>A DMA command consists of source address, destination address, source transfer length, and destination transfer length. It is important that the parameters are programmed in the exact sequence as described.</p>
DMA_SRC_LEN	0x00000020	32	rw	0x00000000	<p>DMA Source transfer Length Register: This register contains the DMA source transfer length in unit of 4-byte word.</p> <p>A DMA command that consists of source address, destination address, source transfer length, and destination transfer length. It is important that the parameters are programmed in the exact sequence as described.</p>
DMA_DEST_LEN	0x00000024	32	rw	0x00000000	<p>DMA Destination transfer Length Register: This register contains the DMA destination transfer length in unit of 4-byte word.</p> <p>A DMA command that consists of source address, destination address, source transfer length, and destination transfer length is accepted when this register is written to. It is important that the parameters are programmed in the exact sequence as described.</p>
ROM_SHADOW	0x00000028	32	wo	0x00000000	<p>ROM Shadow Register: This register defines ROM shadow</p>
MULTIBOOT_ADDR	0x0000002C	32	rw	0x00000000	<p>MULTI Boot Addr Pointer Register: This register defines multi-boot address pointer. This register is power on reset only used to remember multi-boot address pointer set by previous boot.</p>

Register Name	Address	Width	Type	Reset Value	Description
SW_ID	0x00000030	32	mixed	0x00000000	Software ID Register: This register defines PS boot software ID. It will be used by firmware and software to perform consistent check for subsequent PS software and PL image loads. It is both readable and writeable after reset, and it becomes read only after system has entered user mode (bit 15 of reg 0x000 is set).
UNLOCK	0x00000034	32	rw	0x00000000	Unlock Register: This register is used to protect the DEVCI configuration registers from ROM code corruption. The boot ROM will unlock the DEVCI by writing 0x757BDF0D to this register. Writing anything other than the unlock word to this register will cause an illegal access state and make the DEVCI inaccessible until a system reset occurs.
MCTRL	0x00000080	32	mixed	x	Miscellaneous control Register: This register contains miscellaneous controls.
XADCIF_CFG	0x00000100	32	rw	0x00001114	XADC Interface Configuration Register : This register configures the XADC Interface operation
XADCIF_INT_STS	0x00000104	32	mixed	0x00000200	XADC Interface Interrupt Status Register : This register contains the interrupt status flags of the XADC interface block. All register bits are clear on write by writing 1s to those bits, however the register bits will only be cleared if the condition that sets the interrupt flag is no longer true. Note that individual status bits will be set if the corresponding condition is satisfied regardless of whether the interrupt mask bit in 0x108 is set. However, external interrupt will only be generated if an interrupt status flag is set and the corresponding mask bit is not set

Register Name	Address	Width	Type	Reset Value	Description
XADCIF_INT_MASK	0x00000108	32	rw	0xFFFFFFFF	XADC Interface Interrupt Mask Register : This register contains the interrupt mask information. Set a bit to 1 to mask the interrupt generation from the corresponding interrupting source in 0x104
XADCIF_MSTS	0x0000010C	32	ro	0x00000500	XADC Interface miscellaneous Status Register : This register contains miscellaneous status of the XADC Interface
XADCIF_CMDFIFO	0x00000110	32	wo	0x00000000	XADC Interface Command FIFO Register : This address is the entry point to the command FIFO. Commands get push into the FIFO when there is a write to this address
XADCIF_RDFIFO	0x00000114	32	ro	0x00000000	XADC Interface Data FIFO Register : This address is the exit point of the read data FIFO. Read data is returned when there is a read from this address
XADCIF_MCTL	0x00000118	32	rw	0x00000010	XADC Interface Miscellaneous Control Register : This register provides miscellaneous control of the XADC Interface.

Register ([devcfg](#)) CTRL

Name	CTRL
Relative Address	0x00000000
Absolute Address	0xF8007000
Width	32 bits
Access Type	mixed
Reset Value	0x0C000000
Description	Control Register : This register defines basic control registers. Some of the register bits can be locked by control bits in the LOCK Register 0x004.

Register CTRL Details

Field Name	Bits	Type	Reset Value	Description
FORCE_RST	31	rw	0x0	Force the PS into secure lockdown. The secure lockdown state can only be cleared by issuing a PS_POR_B reset
PCFG_PROG_B	30	rw	0x0	Program Signal used to reset the PL. It acts as the PROG_B signal in the PL.
PCFG_POR_CNT_4K	29	rw	0x0	This register controls which POR timer the PL will use for power-up. 0 - Use 64k timer 1 - Use 4k timer
reserved	28	rw	0x0	Reserved
PCAP_PR	27	rw	0x1	After the initial configuration of the PL, a partial reconfiguration can be performed using either the ICAP or PCAP interface. These interfaces are mutually exclusive and cannot be used simultaneously. Switching between ICAP and PCAP is possible but users should ensure that no commands or data are being transmitted or received before changing interfaces. Failure to do this could lead to unexpected behavior. This bit selects between ICAP and PCAP for PL reconfiguration. 0 - ICAP is selected for reconfiguration 1 - PCAP is selected for reconfiguration
PCAP_MODE	26	rw	0x1	This bit enables the PCAP interface
QUARTER_PCAP_RATE_EN (PCAP_RATE_EN)	25	rw	0x0	This bit is used to reduce the PCAP data transmission to once every 4 clock cycles. This bit MUST be set when the AES engine is being used to decrypt configuration data for either the PS or PL. Setting this bit for non-encrypted PCAP data transmission is allowed but not recommended. 0 - PCAP data transmitted every clock cycle 1 - PCAP data transmitted every 4th clock cycle (must be used for encrypted data)
MULTIBOOT_EN	24	rw	0x0	This bit enables multi-boot out of reset. This bit is only cleared by a PS_POR_B reset, 0 - Boot from default boot image base address 1 - Boot from multi-boot offset address

Field Name	Bits	Type	Reset Value	Description
JTAG_CHAIN_DIS	23	rw	0x0	This bit is used to disable the JTAG scan chain. The primary purpose is to protect the PL from unwanted JTAG accesses. The JTAG connection to the PS DAP and PL TAP will be disabled when this bit is set.
reserved	22:16	rw	0x0	Reserved
USER_MODE	15	wo	0x0	Indicates which mode the CPU is operating in 0 - CPU is running in ROM Mode 1 - CPU is running in User Mode *This bit is sticky and is set by the boot ROM
reserved	14	rw	0x0	Reserved - always write with 1
reserved	13	rw	0x0	Reserved - always write with 1
PCFG_AES_FUSE	12	rw	0x0	(Lockable, see 0x004, bit 4) This bit is used to select the AES key source 0 - BBRAM key 1 - eFuse key User access to this bit is restricted. The boot ROM will make the key selection and lock this bit during the initial boot sequence. This bit is only cleared by PS_POR_B reset.
PCFG_AES_EN	11:9	rw	0x0	(Lockable, see 0x004, bit 3) This bit enables the AES engine within the PL. The three bits need to be either all 0's or 1's, any inconsistency will lead to security lockdown. 000 - Disable AES engine 111 - Enable AES engine All others - Secure lockdown User access to this bit is restricted. The boot ROM will enable the AES engine for secure boot and will always lock this bit before passing control to user code. This bit is only cleared by PS_POR_B reset.
SEU_EN	8	rw	0x0	(Lockable, see 0x004, bit 2) This bit enables an automatic lockdown of the PS when a PL SEU is detected. 0 - Ignore SEU signal from PL 1 - Initiate secure lockdown when SEU signal received from PL This bit is sticky, once set it can only be cleared with a PS_POR_B reset.

Field Name	Bits	Type	Reset Value	Description
SEC_EN	7	ro	0x0	(Lockable, see 0x004, bit 1) This bit is used to indicate if the PS has been booted securely. 0 - PS was not booted securely 1 - PS was booted securely User access to this bit is restricted. The boot ROM will set this bit when a secure boot is initiated and will always lock the bit before passing control to user code. This bit is only cleared by PS_POR_B reset.
SPNIDEN	6	rw	0x0	(Lockable, see 0x004, bit 0) Secure Non-Invasive Debug Enable 0 - Disable 1 - Enable
SPIDEN	5	rw	0x0	(Lockable, see 0x004, bit 0) Secure Invasive Debug Enable 0 - Disable 1 - Enable
NIDEN	4	rw	0x0	(Lockable, see 0x004, bit 0) Non-Invasive Debug Enable 0 - Disable 1 - Enable
DBGEN	3	rw	0x0	(Lockable, see 0x004, bit 0) Invasive Debug Enable 0 - Disable 1 - Enable
DAP_EN	2:0	rw	0x0	(Lockable, see 0x004, bit 0) These bits will enable the ARM DAP. 111 - ARM DAP Enabled Others - ARM DAP will be bypassed

Register ([devcfg](#)) LOCK

Name	LOCK
Relative Address	0x00000004
Absolute Address	0xF8007004
Width	32 bits
Access Type	mixed
Reset Value	0x00000000

Description This register defines LOCK register used to lock changes in the Control Register 0x000 after configuration. All those LOCK register is set only register. The only way to clear those registers is power on reset signal.

Register LOCK Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:5	rw	0x0	Reserved
AES_FUSE_LOCK	4	rwso	0x0	<p>This bit locks the PCFG_AES_FUSE bit (CTRL[12]).</p> <p>0 - Open</p> <p>1 - Locked</p> <p>User access to this bit is restricted, the boot ROM will always set this bit prior to handing control over to user code.</p> <p>This bit is only cleared by a PS_POR_B reset.</p>
AES_EN_LOCK (AES_EN)	3	rwso	0x0	<p>This bit locks the PCFG_AES_EN bits (CTRL[11:9]).</p> <p>0 - Open</p> <p>1 - Locked</p> <p>User access to this bit is restricted, the boot ROM will always set this bit prior to handing control over to user code.</p> <p>This bit is only cleared by a PS_POR_B reset.</p>
SEU_LOCK (SEU)	2	rwso	0x0	<p>This bit locks the SEU_EN bit (CTRL[8]).</p> <p>0 - Open</p> <p>1 - Locked</p> <p>This bit is only cleared by a PS_POR_B reset.</p>
SEC_LOCK (SEC)	1	rwso	0x0	<p>This bit locks the SEC_EN bit (CTRL[7]).</p> <p>0 - Open</p> <p>1 - Locked</p> <p>User access to this bit is restricted, the boot ROM will always set this bit prior to handing control over to user code.</p> <p>This bit is only cleared by a PS_POR_B reset.</p>
DBG_LOCK (DBG)	0	rwso	0x0	<p>This bit locks the debug enable bits, SPNIDEN, SPIDEN, NIDEN, DBGEN, DAP_EN (CTRL[6:0]).</p> <p>0 - Open</p> <p>1 - Locked</p> <p>DBG_LOCK should only be used to prevent the debug access from being enabled. If DBG_LOCK is set and a soft-reset is issued, then the DAP_EN bits in the CTRL register (0x000) cannot be enabled until a power-on-reset is performed.</p> <p>This bit is only cleared by a PS_POR_B reset.</p>

Register ([devcfg](#)) CFG

Name	CFG
Relative Address	0x00000008
Absolute Address	0xF8007008
Width	32 bits
Access Type	rw
Reset Value	0x0000050B
Description	Configuration Register : This register contains configuration information for the AXI transfers, and other general setup.

Register CFG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:12	rw	0x0	Reserved
RFIFO_TH	11:10	rw	0x1	These two bits define Rx FIFO level that sets interrupt flag 00 - One fourth full for read 01 - Half full for read 10 - Three fourth full for read 11 - Full for read(User could use this signal to trigger interrupt when read FIFO overflow)
WFIFO_TH	9:8	rw	0x1	These two bits define Tx FIFO level that sets interrupt flag 00 - One fourth empty for write 01 - Half empty for write 10 - Three fourth empty for write 11 - Empty for write
RCLK_EDGE	7	rw	0x0	Read data active clock edge 0 - Falling edge 1 - Rising edge
WCLK_EDGE	6	rw	0x0	Write data active clock edge 0 - Falling edge 1 - Rising edge
DISABLE_SRC_INC	5	rw	0x0	Disable automatic DMA AXI source address increment, if set, to allow AXI read from a keyhole address
DISABLE_DST_INC	4	rw	0x0	Disable automatic DMA AXI destination address increment, if set, to allow AXI read from a keyhole address
reserved	3	rw	0x1	Reserved. Do not modify.

Field Name	Bits	Type	Reset Value	Description
reserved	2	rw	0x0	Reserved. Do not modify.
RDLEN	1	rw	0x1	AXI read burst length 0 - burst-of-8 1 - burst-of-16
WRLEN	0	rw	0x1	AXI write burst length 0 - burst-of-8 1 - burst-of-16

Register ([devcfg](#)) INT_STS

Name	INT_STS
Relative Address	0x0000000C
Absolute Address	0xF800700C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	<p>Interrupt Status Register : This register contains interrupt status flags.</p> <p>All register bits are clear on write by writing 1s to those bits, however the register bits will only be cleared if the condition that sets the interrupt flag is no longer true.</p> <p>Note that individual status bits will be set if the corresponding condition is satisfied regardless of whether the interrupt mask bit in 0x010 is set.</p> <p>However, external interrupt will only be generated if an interrupt status flag is set and the corresponding mask bit is not set</p>

Register INT_STS Details

Field Name	Bits	Type	Reset Value	Description
PSS_GTS_USR_B_INT	31	wtc	0x0	Tri-state IO during HIZ, both edges
PSS_FST_CFG_B_INT	30	wtc	0x0	First configuration done, both edges
PSS_GPWRDWN_B_INT	29	wtc	0x0	Global power down, both edges
PSS_GTS_CFG_B_INT	28	wtc	0x0	Tri-state IO during configuration, both edges
PSS_CFG_RESET_B_INT	27	wtc	0x0	PL configuration reset, both edges
reserved	26:24	rw	0x0	Reserved
AXI_WTO_INT (IXR_AXI_WTO)	23	wtc	0x0	AXI write address, data or response time out. AXI write is taking longer than expected (> 6144 cpu_1x clock cycles), this can be an indication of starvation

Field Name	Bits	Type	Reset Value	Description
AXI_WERR_INT (IXR_AXI_WERR)	22	wtc	0x0	AXI write response error
AXI_RTO_INT (IXR_AXI_RTO)	21	wtc	0x0	AXI read address or response time out. AXI read is taking longer than expected (> 2048 cpu_1x clock cycles), this can be an indication of starvation
AXI_RERR_INT (IXR_AXI_RERR)	20	wtc	0x0	AXI read response error
reserved	19	rw	0x0	Reserved
RX_FIFO_OV_INT (IXR_RX_FIFO_OV)	18	wtc	0x0	This bit is used to indicate that RX FIFO overflows. Incoming read data from PCAP will be dropped and the DEVCI DMA may enter an unrecoverable state .
WR_FIFO_LVL_INT (IXR_WR_FIFO_LVL)	17	wtc	0x0	Tx FIFO level < threshold, see reg 0x008
RD_FIFO_LVL_INT (IXR_RD_FIFO_LVL)	16	wtc	0x0	Rx FIFO level >= threshold, see reg 0x008
DMA_CMD_ERR_INT (IXR_DMA_CMD_ERR)	15	wtc	0x0	Illegal DMA command
DMA_Q_OV_INT (IXR_DMA_Q_OV)	14	wtc	0x0	DMA command queue overflows
DMA_DONE_INT (IXR_DMA_DONE)	13	wtc	0x0	This bit is used to indicate a DMA command is done. The bit is set either as soon as DMA is done (PCAP may not be finished) or both DMA and PCAP are done.
D_P_DONE_INT (IXR_D_P_DONE)	12	wtc	0x0	Both DMA and PCAP transfers are done for intermediate and final transfers.
P2D_LEN_ERR_INT (IXR_P2D_LEN_ERR)	11	wtc	0x0	Inconsistent PCAP to DMA transfer length error
reserved	10:7	rw	0x0	Reserved
PCFG_HMAC_ERR_I NT (IXR_PCFG_HMAC_E RR)	6	wtc	0x0	HMAC error from PL
PCFG_SEU_ERR_INT (IXR_PCFG_SEU_ERR)	5	wtc	0x0	SEU status from PL
PCFG_POR_B_INT (IXR_PCFG_POR_B)	4	wtc	0x0	PL POR Indication

Field Name	Bits	Type	Reset Value	Description
PCFG_CFG_RST_INT (IXR_PCFG_CFG_RST)	3	wtc	0x0	Signal used to indicate PL under reset
PCFG_DONE_INT (IXR_PCFG_DONE)	2	wtc	0x0	DONE signal from PL indicating that programming is complete and PL is in user mode.
PCFG_INIT_PE_INT (IXR_PCFG_INIT_PE)	1	wtc	0x0	Triggered on the positive edge of the PL INIT signal
PCFG_INIT_NE_INT (IXR_PCFG_INIT_NE)	0	wtc	0x0	Triggered on the negative edge of the PL INIT signal

Register ([devcfg](#)) INT_MASK

Name	INT_MASK
Relative Address	0x00000010
Absolute Address	0xF8007010
Width	32 bits
Access Type	rw
Reset Value	0xFFFFFFFF
Description	<p>Interrupt Mask Register: This register contains interrupt mask information.</p> <p>Set a bit to 1 to mask the interrupt generation from the corresponding interrupting source in Interrupt Status Register 0x00C.</p>

Register INT_MASK Details

Field Name	Bits	Type	Reset Value	Description
M_PSS_GTS_USR_B_INT	31	rw	0x1	Interrupt mask for tri-state IO during HIZ, both edges
M_PSS_FST_CFG_B_INT	30	rw	0x1	Interrupt mask for first config done, both edges
M_PSS_GPWRDWN_B_INT	29	rw	0x1	Interrupt mask for global power down, both edges
M_PSS_GTS_CFG_B_INT	28	rw	0x1	Interrupt mask for tri-state IO in config, both edges
M_PSS_CFG_RESET_B_INT	27	rw	0x1	Interrupt mask for config reset, both edges
reserved	26:24	rw	0x7	Reserved
M_AXI_WTO_INT (IXR_AXI_WTO)	23	rw	0x1	Interrupt mask for AXI write time out interrupt
M_AXI_WERR_INT (IXR_AXI_WERR)	22	rw	0x1	Interrupt mask for AXI write response error interrupt

Field Name	Bits	Type	Reset Value	Description
M_AXI_RTO_INT (IXR_AXI_RTO)	21	rw	0x1	Interrupt mask for AXI read time out interrupt
M_AXI_RERR_INT (IXR_AXI_RERR)	20	rw	0x1	Interrupt mask for AXI read response error interrupt
reserved	19	rw	0x1	Reserved
M_RX_FIFO_OV_INT (IXR_RX_FIFO_OV)	18	rw	0x1	Interrupt mask for Rx FIFO overflow interrupt
M_WR_FIFO_LVL_INT (IXR_WR_FIFO_LVL)	17	rw	0x1	Interrupt mask for Tx FIFO level < threshold interrupt
M_RD_FIFO_LVL_INT (IXR_RD_FIFO_LVL)	16	rw	0x1	Interrupt mask for Rx FIFO level > threshold interrupt
M_DMA_CMD_ERR_INT (IXR_DMA_CMD_ERR)	15	rw	0x1	Interrupt mask for illegal DMA command interrupt
M_DMA_FIFO_OV_INT (IXR_DMA_Q_OV)	14	rw	0x1	Interrupt mask for DMA command FIFO overflows
M_DMA_DONE_INT (IXR_DMA_DONE)	13	rw	0x1	Interrupt mask for DMA command done interrupt
M_D_P_DONE_INT (IXR_D_P_DONE)	12	rw	0x1	Interrupt mask for DMA and PCAP done interrupt
M_P2D_LEN_ERR_INT (IXR_P2D_LEN_ERR)	11	rw	0x1	Interrupt mask Inconsistent xfer length error interrupt
reserved	10:7	rw	0xF	Reserved
M_PCFG_HMAC_ERR_INT (IXR_PCFG_HMAC_ERR)	6	rw	0x1	Interrupt mask for HMAC error
M_PCFG_SEU_ERR_INT (IXR_PCFG_SEU_ERR)	5	rw	0x1	Interrupt mask for PCFG_SEU_ERR interrupt
M_PCFG_POR_B_INT (IXR_PCFG_POR_B)	4	rw	0x1	Interrupt mask for PCFG_POR_B Interrupt
M_PCFG_CFG_RST_INT (IXR_PCFG_CFG_RST)	3	rw	0x1	Interrupt mask for PCFG_CFG_RESET interrupt

Field Name	Bits	Type	Reset Value	Description
M_PCFG_DONE_INT (IXR_PCFG_DONE)	2	rw	0x1	Interrupt mask for PCFG_DONE interrupt
M_PCFG_INIT_PE_INT (IXR_PCFG_INIT_PE)	1	rw	0x1	Interrupt mask for PCFG_INIT_PE interrupt
M_PCFG_INIT_NE_INT (IXR_PCFG_INIT_NE)	0	rw	0x1	Interrupt mask for PCFG_INIT_NE interrupt

Register ([devcfg](#)) STATUS

Name	STATUS
Relative Address	0x00000014
Absolute Address	0xF8007014
Width	32 bits
Access Type	mixed
Reset Value	0x40000820
Description	Status Register: This register contains miscellaneous status.

Register STATUS Details

Field Name	Bits	Type	Reset Value	Description
DMA_CMD_Q_F	31	ro	0x0	DMA command queue full, if set
DMA_CMD_Q_E	30	ro	0x1	DMA command queue empty, if set
DMA_DONE_CNT	29:28	clon wr	0x0	Number of completed DMA transfers that have not been acknowledged by software: 00 - all finished transfers have been acknowledged 01 - one finished transfer outstanding 10 - two finished transfers outstanding 11 - three or more finished transfers outstanding A finished transfer is acknowledged by clearing the interrupt status flag, i.e., bit 9 of the interrupt status register 0x00C. This count is cleared by writing a 1 to either bit location.
reserved	27:25	rw	0x0	Reserved
RX_FIFO_LVL	24:20	ro	0x0	This register is used to indicate how many valid 32-Bit words in the Rx FIFO, max. is 31
reserved	19	rw	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
TX_FIFO_LVL	18:12	ro	0x0	This register is used to indicate how many valid 32-Bit words in the Tx FIFO, max. is 127
PSS_GTS_USR_B	11	ro	0x1	Tri-state IO during HIZ, active low
PSS_FST_CFG_B	10	ro	0x0	First PL configuration done, active low.
PSS_GPWRDWN_B	9	ro	0x0	Global power down, active low
PSS_GTS_CFG_B	8	ro	0x0	Tri-state IO during config, active low
SECURE_RST	7	ro	0x0	This bit is used to indicate a secure lockdown. Can only be cleared by a PS_POR_B reset.
ILLEGAL_APB_ACCE SS	6	ro	0x0	Indicates the UNLOCK register was not written with the correct unlock word. If set all secure boot features will be disabled, the DAP will be disabled and writing to the DEVCI registers will be disabled. The illegal access mode can only be cleared with a PS_POR_B reset.
PSS_CFG_RESET_B	5	ro	0x1	PL configuration reset, active low.
PCFG_INIT	4	ro	0x0	PL INIT signal, indicates when housecleaning is done and the PL is ready to receive PCAP data. Positive and negative edges of the signal generate maskable interrupts in 0x00C.
EFUSE_BBRAM_KEY_ DISABLE (EFUSE_SW_RESERVE)	3	ro	0x0	When this eFuse is blown, the BBRAM AES key is disabled. If the device is booted securely, the eFuse key must be used.
EFUSE_SEC_EN	2	ro	0x0	When this eFuse is blown, the Zynq device must boot securely and use the eFuse as the AES key source. Non-secure boot will cause a security lockdown.
EFUSE_JTAG_DIS	1	ro	0x0	When this eFuse is blown, the ARM DAP controller is permanently set in bypass mode. Any attempt to activate the DAP will cause a security lockdown.
reserved	0	ro	0x0	Reserved. Do not modify.

Register ([devcfg](#)) DMA_SRC_ADDR

Name	DMA_SRC_ADDR
Relative Address	0x00000018
Absolute Address	0xF8007018
Width	32 bits

Access Type	rw
Reset Value	0x00000000
Description	<p>DMA Source address Register: This register contains the source address for DMA transfer.</p> <p>A DMA command consists of source address, destination address, source transfer length, and destination transfer length.</p> <p>It is important that the parameters are programmed in the exact sequence as described</p>

Register DMA_SRC_ADDR Details

Field Name	Bits	Type	Reset Value	Description
ADDR	31:0	rw	0x0	Source address for DMA transfer of AXI read

Register ([devcfg](#)) DMA_DST_ADDR

Name	DMA_DST_ADDR
Software Name	DMA_DEST_ADDR
Relative Address	0x0000001C
Absolute Address	0xF800701C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	<p>DMA Destination address Register: This register contains the destination address for DMA transfer.</p> <p>A DMA command consists of source address, destination address, source transfer length, and destination transfer length.</p> <p>It is important that the parameters are programmed in the exact sequence as described.</p>

Register DMA_DST_ADDR Details

Field Name	Bits	Type	Reset Value	Description
ADDR	31:0	rw	0x0	Destination address for DMA transfer of AXI write

Register ([devcfg](#)) DMA_SRC_LEN

Name	DMA_SRC_LEN
Relative Address	0x00000020
Absolute Address	0xF8007020
Width	32 bits

Access Type	rw
Reset Value	0x00000000
Description	<p>DMA Source transfer Length Register: This register contains the DMA source transfer length in unit of 4-byte word.</p> <p>A DMA command that consists of source address, destination address, source transfer length, and destination transfer length.</p> <p>It is important that the parameters are programmed in the exact sequence as described.</p>

Register DMA_SRC_LEN Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:27	rw	0x0	Reserved
LEN (DMA_LEN)	26:0	rw	0x0	Up to 512MB data

Register ([devcfg](#)) DMA_DEST_LEN

Name	DMA_DEST_LEN
Relative Address	0x00000024
Absolute Address	0xF8007024
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	<p>DMA Destination transfer Length Register: This register contains the DMA destination transfer length in unit of 4-byte word.</p> <p>A DMA command that consists of source address, destination address, source transfer length, and destination transfer length is accepted when this register is written to.</p> <p>It is important that the parameters are programmed in the exact sequence as described.</p>

Register DMA_DEST_LEN Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:27	rw	0x0	Reserved
LEN (DMA_LEN)	26:0	rw	0x0	Up to 512MB data

Register ([devcfg](#)) ROM_SHADOW

Name	ROM_SHADOW
Relative Address	0x00000028
Absolute Address	0xF8007028
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	ROM Shadow Register: This register defines ROM shadow

Register ROM_SHADOW Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:0	wo	0x0	Reserved. Do not modify.

Register ([devcfg](#)) MULTIBOOT_ADDR

Name	MULTIBOOT_ADDR
Relative Address	0x0000002C
Absolute Address	0xF800702C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	MULTI Boot Addr Pointer Register: This register defines multi-boot address pointer. This register is power on reset only used to remember multi-boot address pointer set by previous boot.

Register MULTIBOOT_ADDR Details

Field Name	Bits	Type	Reset Value	Description
MULTIBOOT_ADDR	31:0	rw	0x0	Multi-Boot offset address

Register ([devcfg](#)) SW_ID

Name	SW_ID
Relative Address	0x00000030
Absolute Address	0xF8007030
Width	32 bits
Access Type	mixed
Reset Value	0x00000000

Description Software ID Register: This register defines PS boot software ID. It will be used by firmware and software to perform consistent check for subsequent PS software and PL image loads. It is both readable and writeable after reset, and it becomes read only after system has entered user mode (bit 15 of reg 0x000 is set).

Register SW_ID Details

Field Name	Bits	Type	Reset Value	Description
SW_ID	31:0	rw,ro	0x0	Software ID

Register ([devcfg](#)) UNLOCK

Name UNLOCK

Relative Address 0x00000034

Absolute Address 0xF8007034

Width 32 bits

Access Type rw

Reset Value 0x00000000

Description Unlock Register: This register is used to protect the DEVCI configuration registers from ROM code corruption.
The boot ROM will unlock the DEVCI by writing 0x757BDF0D to this register.
Writing anything other than the unlock word to this register will cause an illegal access state and make the DEVCI inaccessible until a system reset occurs.

Register UNLOCK Details

Field Name	Bits	Type	Reset Value	Description
UNLOCK	31:0	rw	0x0	Unlock value.

Register ([devcfg](#)) MCTRL

Name MCTRL

Relative Address 0x00000080

Absolute Address 0xF8007080

Width 32 bits

Access Type mixed

Reset Value x

Description Miscellaneous control Register: This register contains miscellaneous controls.

Register MCTRL Details

Field Name	Bits	Type	Reset Value	Description
PS_VERSION	31:28	ro	x	Version ID for silicon 0x0 = 1.0 Silicon 0x1 = 2.0 Silicon 0x2 = 3.0 Silicon
reserved	27:24	rw	0x0	Reserved
reserved	23	rw	0x1	Reserved - always write with 1
reserved	22:9	rw	0x0	Reserved
PCFG_POR_B	8	ro	0x0	PL POR_B signal used to determine power-up status of PL.
reserved	7:5	rw	0x0	Reserved
INT_PCAP_LPBK (PCAP_LPBK)	4	rw	0x0	Internal PCAP loopback, if set
reserved	3:2	rw	0x0	Reserved
reserved	1	rw	0x0	Reserved - always write with 0
reserved	0	rw	0x0	Reserved - always write with 0

Register ([devcfg](#)) XADCIF_CFG

Name	XADCIF_CFG
Relative Address	0x00000100
Absolute Address	0xF8007100
Width	32 bits
Access Type	rw
Reset Value	0x00001114
Description	XADC Interface Configuration Register : This register configures the XADC Interface operation

Register XADCIF_CFG Details

Field Name	Bits	Type	Reset Value	Description
ENABLE	31	rw	0x0	Enable PS access of the XADC, if set
reserved	30:24	rw	0x0	Reserved
CFIFOTH	23:20	rw	0x0	Command FIFO level threshold. Interrupt status flag is set if the FIFO level is less than or equal to the threshold

Field Name	Bits	Type	Reset Value	Description
DFIFOTH	19:16	rw	0x0	Data FIFO level threshold. Interrupt status flag is set if FIFO level is greater than the threshold
reserved	15:14	rw	0x0	Reserved
WEDGE	13	rw	0x0	Write launch edge : 0 - Falling edge 1 - Rising edge
REDGE	12	rw	0x1	Read capture edge : 0 - Falling edge 1 - Rising edge
reserved	11:10	rw	0x0	Reserved
TCKRATE	9:8	rw	0x1	XADC clock frequency control. The base frequency is pcap_2x clock which has a nominal frequency of 200 MHz. 00 - 1/2 of pcap_2x clock frequency 01 - 1/4 of pcap_2x clock frequency 10 - 1/8 of pcap_2x clock frequency 11 - 1/16 of pcap_2x clock frequency
reserved	7:5	rw	0x0	Reserved
IGAP	4:0	rw	0x14	Minimum idle gap between successive commands.

Register ([devcfg](#)) XADCIF_INT_STS

Name	XADCIF_INT_STS
Relative Address	0x00000104
Absolute Address	0xF8007104
Width	32 bits
Access Type	mixed
Reset Value	0x00000200
Description	<p>XADC Interface Interrupt Status Register : This register contains the interrupt status flags of the XADC interface block.</p> <p>All register bits are clear on write by writing 1s to those bits, however the register bits will only be cleared if the condition that sets the interrupt flag is no longer true.</p> <p>Note that individual status bits will be set if the corresponding condition is satisfied regardless of whether the interrupt mask bit in 0x108 is set.</p> <p>However, external interrupt will only be generated if an interrupt status flag is set and the corresponding mask bit is not set</p>

Register XADCIF_INT_STS Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	rw	0x0	Reserved
CFIFO_LTH	9	wtc	0x1	Command FIFO level less than or equal to the threshold (see register 0x100).
DFIFO_GTH	8	wtc	0x0	Data FIFO level greater than threshold (see register 0x100).
OT	7	wtc	0x0	Over temperature alarm from XADC. This is a latched version of the raw signal which is also available in register 0x10C
ALM	6:0	wtc	0x0	Alarm signals from XADC. These are latched version of the raw input alarm signals which are also available in register 0x10C

Register ([devcfg](#)) XADCIF_INT_MASK

Name	XADCIF_INT_MASK
Relative Address	0x00000108
Absolute Address	0xF8007108
Width	32 bits
Access Type	rw
Reset Value	0xFFFFFFFF
Description	XADC Interface Interrupt Mask Register : This register contains the interrupt mask information. Set a bit to 1 to mask the interrupt generation from the corresponding interrupting source in 0x104

Register XADCIF_INT_MASK Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	rw	0x3FFFFFFF	Reserved
M_CFIFO_LTH	9	rw	0x1	Interrupt mask for command FIFO level threshold interrupt.
M_DFIFO_GTH	8	rw	0x1	Interrupt mask Data FIFO level greater than threshold interrupt.
M_OT	7	rw	0x1	Interrupt mask for over temperature alarm interrupt
M_ALM	6:0	rw	0x7F	Interrupt mask for alarm signals from XADC.

Register ([devcfg](#)) XADCIF_MSTS

Name	XADCIF_MSTS
Relative Address	0x0000010C
Absolute Address	0xF800710C
Width	32 bits
Access Type	ro
Reset Value	0x00000500
Description	XADC Interface miscellaneous Status Register : This register contains miscellaneous status of the XADC Interface

Register XADCIF_MSTS Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:20	ro	0x0	Reserved
CFIFO_LVL	19:16	ro	0x0	Command FIFO level.
DFIFO_LVL	15:12	ro	0x0	Data FIFO level.
CFIFO_F	11	ro	0x0	Command FIFO full.
CFIFO_E	10	ro	0x1	Command FIFO empty.
DFIFO_F	9	ro	0x0	Data FIFO full.
DFIFO_E	8	ro	0x1	Data FIFO empty.
OT	7	ro	0x0	Raw over temperature alarm from the XADC. Latched version of the signal is available in the interrupt status register.
ALM	6:0	ro	0x0	Raw alarm signals from the XADC. Latched version of the signals are available in the interrupt status register.

Register ([devcfg](#)) XADCIF_CMDFIFO

Name	XADCIF_CMDFIFO
Relative Address	0x00000110
Absolute Address	0xF8007110
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	XADC Interface Command FIFO Register : This address is the entry point to the command FIFO. Commands get push into the FIFO when there is a write to this address

Register XADCIF_CMDFIFO Details

Field Name	Bits	Type	Reset Value	Description
CMD	31:0	wo	0x0	32-bit command.

Register ([devcfg](#)) XADCIF_RDFIFO

Name	XADCIF_RDFIFO
Relative Address	0x00000114
Absolute Address	0xF8007114
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	XADC Interface Data FIFO Register : This address is the exit point of the read data FIFO. Read data is returned when there is a read from this address

Register XADCIF_RDFIFO Details

Field Name	Bits	Type	Reset Value	Description
RDDATA	31:0	ro	0x0	32-bit read data.

Register ([devcfg](#)) XADCIF_MCTL

Name	XADCIF_MCTL
Relative Address	0x00000118
Absolute Address	0xF8007118
Width	32 bits
Access Type	rw
Reset Value	0x00000010
Description	XADC Interface Miscellaneous Control Register : This register provides miscellaneous control of the XADC Interface.

Register XADCIF_MCTL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:5	rw	0x0	Reserved
RESET	4	rw	0x1	This bit will reset the communication channel between the PS and XADC. If set, the PS-XADC communication channel will remain in reset until a 0 is written to this bit.

Field Name	Bits	Type	Reset Value	Description
reserved	3:1	rw	0x0	Reserved
reserved	0	rw	0x0	Reserved - always write with 0

B.17 DMA Controller (dmac)

Module Name	DMA Controller (dmac)
Software Name	XDMAPS
Base Address	0xF8004000 dmac0_ns 0xF8003000 dmac0_s
Description	direct memory access controller, PL330 This address space is used when the DMAC is in non-secure mode.
Version	R1P1
Doc Version	1.0
Vendor Info	ARM PL330

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
DSR	0x00000000	32	mixed	0x00000000	DMA manager status register
DPC	0x00000004	32	mixed	0x00000000	DMA Program Counter Register
INTEN	0x00000020	32	mixed	0x00000000	Interrupt enable register
INT_EVENT_RIS	0x00000024	32	mixed	0x00000000	Event interrupt raw status register
INTMIS	0x00000028	32	mixed	0x00000000	Interrupt Status Register
INTCLR	0x0000002C	32	mixed	0x00000000	Interrupt Clear Register
FSRD	0x00000030	32	mixed	0x00000000	Fault Status DMA Manager Register
FSRC	0x00000034	32	mixed	0x00000000	Fault Status DMA Channel Register
FTRD	0x00000038	32	mixed	0x00000000	Fault Type DMA Manager Register
FTR0	0x00000040	32	mixed	0x00000000	Default type for DMA channel 0
FTR1	0x00000044	32	mixed	0x00000000	Default type for DMA channel 1
FTR2	0x00000048	32	mixed	0x00000000	Default type for DMA channel 2
FTR3	0x0000004C	32	mixed	0x00000000	Default type for DMA channel 3
FTR4	0x00000050	32	mixed	0x00000000	Default type for DMA channel 4
FTR5	0x00000054	32	mixed	0x00000000	Default type for DMA channel 5
FTR6	0x00000058	32	mixed	0x00000000	Default type for DMA channel 6
FTR7	0x0000005C	32	mixed	0x00000000	Default type for DMA channel 7

Register Name	Address	Width	Type	Reset Value	Description
CSR0	0x00000100	32	mixed	0x00000000	Channel status for DMA channel 0
CPC0	0x00000104	32	mixed	0x00000000	Channel PC for DMA channel 0
CSR1	0x00000108	32	mixed	0x00000000	Channel status for DMA channel 1
CPC1	0x0000010C	32	mixed	0x00000000	Channel PC for DMA channel 1
CSR2	0x00000110	32	mixed	0x00000000	Channel status for DMA channel 2
CPC2	0x00000114	32	mixed	0x00000000	Channel PC for DMA channel 2
CSR3	0x00000118	32	mixed	0x00000000	Channel status for DMA channel 3
CPC3	0x0000011C	32	mixed	0x00000000	Channel PC for DMA channel 3
CSR4	0x00000120	32	mixed	0x00000000	Channel status for DMA channel 4
CPC4	0x00000124	32	mixed	0x00000000	Channel PC for DMA channel 4
CSR5	0x00000128	32	mixed	0x00000000	Channel status for DMA channel 5
CPC5	0x0000012C	32	mixed	0x00000000	Channel PC for DMA channel 5
CSR6	0x00000130	32	mixed	0x00000000	Channel status for DMA channel 6
CPC6	0x00000134	32	mixed	0x00000000	Channel PC for DMA channel 6
CSR7	0x00000138	32	mixed	0x00000000	Channel status for DMA channel 7
CPC7	0x0000013C	32	mixed	0x00000000	Channel PC for DMA channel 7
SAR0	0x00000400	32	mixed	0x00000000	source address for DMA channel 0
DAR0	0x00000404	32	mixed	0x00000000	destination address for DMA channel 0
CCR0	0x00000408	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x00800200	channel control for DMA channel 0
LC0_0	0x0000040C	32	mixed	0x00000000	loop counter 0 for DMA channel 0
LC1_0	0x00000410	32	mixed	0x00000000	loop counter 1 for DMA channel 0
SAR1	0x00000420	32	mixed	0x00000000	source address for DMA channel 1
DAR1	0x00000424	32	mixed	0x00000000	destination address for DMA channel 1

Register Name	Address	Width	Type	Reset Value	Description
CCR1	0x00000428	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x00800200	channel control for DMA channel 1
LC0_1	0x0000042C	32	mixed	0x00000000	loop counter 0 for DMA channel 1
LC1_1	0x00000430	32	mixed	0x00000000	loop counter 1 for DMA channel 1
SAR2	0x00000440	32	mixed	0x00000000	source address for DMA channel 2
DAR2	0x00000444	32	mixed	0x00000000	destination address for DMA channel 2
CCR2	0x00000448	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x00800200	channel control for DMA channel 2
LC0_2	0x0000044C	32	mixed	0x00000000	loop counter 0 for DMA channel 2
LC1_2	0x00000450	32	mixed	0x00000000	loop counter 1 for DMA channel 2
SAR3	0x00000460	32	mixed	0x00000000	source address for DMA channel 3
DAR3	0x00000464	32	mixed	0x00000000	destination address for DMA channel 3
CCR3	0x00000468	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x00800200	channel control for DMA channel 3
LC0_3	0x0000046C	32	mixed	0x00000000	loop counter 0 for DMA channel 3
LC1_3	0x00000470	32	mixed	0x00000000	loop counter 1 for DMA channel 3
SAR4	0x00000480	32	mixed	0x00000000	source address for DMA channel 4
DAR4	0x00000484	32	mixed	0x00000000	destination address for DMA channel 4
CCR4	0x00000488	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x00800200	channel control for DMA channel 4
LC0_4	0x0000048C	32	mixed	0x00000000	loop counter 0 for DMA channel 4

Register Name	Address	Width	Type	Reset Value	Description
LC1_4	0x00000490	32	mixed	0x00000000	loop counter 1 for DMA channel 4
SAR5	0x000004A0	32	mixed	0x00000000	source address for DMA channel 5
DAR5	0x000004A4	32	mixed	0x00000000	destination address for DMA channel 5
CCR5	0x000004A8	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x00800200	channel control for DMA channel 5
LC0_5	0x000004AC	32	mixed	0x00000000	loop counter 0 for DMA channel 5
LC1_5	0x000004B0	32	mixed	0x00000000	loop counter 1 for DMA channel 5
SAR6	0x000004C0	32	mixed	0x00000000	source address for DMA channel 6
DAR6	0x000004C4	32	mixed	0x00000000	destination address for DMA channel 6
CCR6	0x000004C8	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x00800200	channel control for DMA channel 6
LC0_6	0x000004CC	32	mixed	0x00000000	loop counter 0 for DMA channel 6
LC1_6	0x000004D0	32	mixed	0x00000000	loop counter 1 for DMA channel 6
SAR7	0x000004E0	32	mixed	0x00000000	source address for DMA channel 7
DAR7	0x000004E4	32	mixed	0x00000000	destination address for DMA channel 7
CCR7	0x000004E8	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x00800200	channel control for DMA channel 7
LC0_7	0x000004EC	32	mixed	0x00000000	loop counter 0 for DMA channel 7
LC1_7	0x000004F0	32	mixed	0x00000000	loop counter 1 for DMA channel 7
DBGSTATUS	0x00000D00	32	mixed	0x00000000	Debug Status Register
DBGCMD	0x00000D04	32	mixed	0x00000000	Debug Command Register

Register Name	Address	Width	Type	Reset Value	Description
DBGINST0	0x00000D08	32	mixed	0x00000000	debug instruction 0 register, Controls the debug instruction, channel, and thread information for the DMAC.
DBGINST1	0x00000D0C	32	mixed	0x00000000	debug instruction 0 register, Controls the upper bytes of the debug instruction for the DMAC
CR0	0x00000E00	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x001E3071	Configuration Register 0, Provides the status of the tie-off control signals.
CR1	0x00000E04	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x00000074	Configuration Register 1, Provides information about the instruction cache configuration.
CR2	0x00000E08	32	mixed	0x00000000	Configuration Register 2, Provides the value of the boot address that boot_addr[31:0] configures.
CR3	0x00000E0C	32	mixed	0x00000000	Configuration Register 3, Provides the security state of the event-interrupt resources that are initialized when the DMAC exits from reset.
CR4	0x00000E10	32	mixed	0x00000000	Configuration Register 4, Provides the security state of the peripheral request interfaces that is initialized when the DMAC exits from reset.
CRD	0x00000E14	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x07FF7F73	DMA configuration register, Provides information about the configuration of the data buffer, data width, and read and write issuing capability of the DMAC.
WD	0x00000E80	32	mixed	0x00000000	watch dog register, control the watch dog behavior
periph_id_0	0x00000FE0	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x00000030	peripheral identification register 0, Provides information about the configuration and version of the peripheral

Register Name	Address	Width	Type	Reset Value	Description
periph_id_1	0x0000FE4	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x00000013	peripheral identification register 1, The periph_id_1 Register is hard-coded and the fields in the register control the reset value.
periph_id_2	0x0000FE8	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x00000024	peripheral identification register 2, The periph_id_2 Register is hard-coded and the fields in the register control the reset value.
periph_id_3	0x0000FEC	32	mixed	0x00000000	peripheral identification register 3, Provides information about the configuration and version of the peripheral
pcell_id_0	0x0000FF0	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x0000000D	component identification register 0, When concatenated, these four registers return 0xB105F00D.
pcell_id_1	0x0000FF4	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x000000F0	component identification register 0, When concatenated, these four registers return 0xB105F00D.
pcell_id_2	0x0000FF8	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x00000005	component identification register 0, When concatenated, these four registers return 0xB105F00D.
pcell_id_3	0x0000FFC	32	mixed	dmac0_ns: 0x00000000 dmac0_s: 0x000000B1	component identification register 0, When concatenated, these four registers return 0xB105F00D.

Register ([dmac](#)) DSR

Name	DSR
Software Name	DS
Relative Address	0x00000000
Absolute Address	dmac0_ns: 0xF8004000 dmac0_s: 0xF8003000
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	DMA manager status register

Register DSR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	rud	0x0	Reserved, read undefined
DNS	9	sro,ns sraz,n snsro	0x0	Provides the security status of the DMA manager thread: 0 = DMA manager operates in the Secure state 1 = DMA manager operates in the Non-secure state.
Wakeup_event	8:4	sro,ns sraz,n snsro	0x0	When the DMA manager thread executes a DMAWFE instruction, it waits for the following event to occur: b00000 = event[0] b00001 = event[1] b00010 = event[2] . . . b11111 = event[31].
DMA_status	3:0	sro,ns sraz,n snsro	0x0	The operating state of the DMA manager: b0000 = Stopped b0001 = Executing b0010 = Cache miss b0011 = Updating PC b0100 = Waiting for event b0101-b1110 = reserved b1111 = Faulting.

Register ([dmac](#)) DPC

Name	DPC
Relative Address	0x00000004
Absolute Address	dmac0_ns: 0xF8004004 dmac0_s: 0xF8003004
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	DMA Program Counter Register

Register DPC Details

Field Name	Bits	Type	Reset Value	Description
pc_mgr	31:0	sro,ns sraz,n snsro	0x0	Program counter for the DMA manager thread

Register ([dmac](#)) INTEN

Name	INTEN
Relative Address	0x00000020
Absolute Address	dmac0_ns: 0xF8004020 dmac0_s: 0xF8003020
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Interrupt enable register

Register INTEN Details

Field Name	Bits	Type	Reset Value	Description
event_irq_select	31:0	srw,ns sraz,n snsrw	0x0	Program the appropriate bit to control how the DMAC responds when it executes DMASEV: Bit [N] = 0 If the DMAC executes DMASEV for the event-interrupt resource N then the DMAC signals event N to all of the threads. Set bit [N] to 0 if your system design does not use irq[N] to signal an interrupt request. Bit [N] = 1 If the DMAC executes DMASEV for the event-interrupt resource N then the DMAC sets irq[N] HIGH. Set bit [N] to 1 if your system design requires irq[N] to signal an interrupt request.

Register ([dmac](#)) INT_EVENT_RIS

Name	INT_EVENT_RIS
Software Name	ES
Relative Address	0x00000024
Absolute Address	dmac0_ns: 0xF8004024 dmac0_s: 0xF8003024

Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Event interrupt raw status register

Register INT_EVENT_RIS Details

Field Name	Bits	Type	Reset Value	Description
DMASEV_active	31:0	sro,ns sraz,n snsro	0x0	Returns the status of the event-interrupt resources: Bit [N] = 0 Event N is inactive or irq[N] is LOW. Bit [N] = 1 Event N is active or irq[N] is HIGH. Note When the DMAC executes a DMASEV N instruction to send event N, the INTEN Register controls whether the DMAC: signals an interrupt using the appropriate irq sends the event to all of the threads.

Register ([dmac](#)) INTMIS

Name	INTMIS
Software Name	INTSTATUS
Relative Address	0x00000028
Absolute Address	dmac0_ns: 0xF8004028 dmac0_s: 0xF8003028
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Interrupt Status Register

Register INTMIS Details

Field Name	Bits	Type	Reset Value	Description
irq_status	31:0	sro,ns sraz,n snsro	0x0	Provides the status of the interrupts that are active in the DMAC: Bit [N] = 0 Interrupt N is inactive and therefore irq[N] is LOW. Bit [N] = 1 Interrupt N is active and therefore irq[N] is HIGH. Note You must use the INTCLR Register to set bit [N] to 0, see Interrupt Clear Register. Note Bit [N] is 0 if the INTEN Register programs DMASEV to signal an event

Register ([dmac](#)) INTCLR

Name	INTCLR
Relative Address	0x0000002C
Absolute Address	dmac0_ns: 0xF800402C dmac0_s: 0xF800302C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Interrupt Clear Register

Register INTCLR Details

Field Name	Bits	Type	Reset Value	Description
irq_clr	31:0	swo,n ssraz, nsns wo	0x0	Controls the clearing of the irq outputs: Bit [N] = 0 The status of irq[N] does not change. Bit [N] = 1 The DMAC sets irq[N] LOW if the INTEN Register programs the DMAC to signal an interrupt. Otherwise, the status of irq[N] does not change.

Register ([dmac](#)) FSRD

Name	FSRD
Software Name	FSM
Relative Address	0x00000030

Absolute Address	dmac0_ns: 0xF8004030 dmac0_s: 0xF8003030
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Fault Status DMA Manager Register

Register FSRD Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rud	0x0	reserved, read undefined
fs_mgr	0	sro,ns sraz,n snsro	0x0	Provides the fault status of the DMA manager. Read as: 0 = the DMA manager thread is not in the Faulting state 1 = the DMA manager thread is in the Faulting state.

Register ([dmac](#)) FSRC

Name	FSRC
Software Name	FSC
Relative Address	0x00000034
Absolute Address	dmac0_ns: 0xF8004034 dmac0_s: 0xF8003034
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Fault Status DMA Channel Register

Register FSRC Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
fault_status	7:0	sro,ns sraz,n snsro	0x0	Each bit provides the fault status of the corresponding channel. Read as: Bit [N] = 0 No fault is present on DMA channel N. Bit [N] = 1 DMA channel N is in the Faulting or Faulting completing state.

Register ([dmac](#)) FTRD

Name	FTRD
Software Name	FTM
Relative Address	0x00000038
Absolute Address	dmac0_ns: 0xF8004038 dmac0_s: 0xF8003038
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Fault Type DMA Manager Register

Register FTRD Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	rud	0x0	read undefined
dbg_instr	30	sro,ns sraz,n snsro	0x0	If the DMA manager aborts, this bit indicates whether the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface.
reserved	29:17	rud	0x0	read undefined
instr_fetch_err	16	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA manager performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response
reserved	15:6	rud	0x0	read undefined
mgr_evt_err	5	sro,ns sraz,n snsro	0x0	Indicates whether the DMA manager was attempting to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = the DMA manager has appropriate security to execute DMAWFE or DMASEV 1 = a DMA manager thread in the Non-secure state attempted to execute either: DMAWFE to wait for a secure event H18DMASEV to create a secure event or secure interrupt.

Field Name	Bits	Type	Reset Value	Description
dmago_err	4	sro,ns sraz,n snsro	0x0	Indicates whether the DMA manager was attempting to execute DMAGO with inappropriate security permissions: 0 = the DMA manager has appropriate security to execute DMAGO 1 = a DMA manager thread in the Non-secure state attempted to execute DMAGO to create a DMA channel operating in the Secure state.
reserved	3:2	rud	0x0	read undefined
operand_invalid	1	sro,ns sraz,n snsro	0x0	Indicates whether the DMA manager was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand.
undef_instr	0	sro,ns sraz,n snsro	0x0	Indicates whether the DMA manager was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction.

Register ([dmac](#)) FTR0

Name	FTR0
Software Name	FTC0
Relative Address	0x00000040
Absolute Address	dmac0_ns: 0xF8004040 dmac0_s: 0xF8003040
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Default type for DMA channel 0

Register FTR0 Details

Field Name	Bits	Type	Reset Value	Description
lockup_err	31	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel has locked-up because of resource starvation: 0 = DMA channel has adequate resources 1 = DMA channel has locked-up because of insufficient resources. This fault is an imprecise abort.
dbg_instr	30	sro,ns sraz,n snsro	0x0	If the DMA channel aborts, this bit indicates whether the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface. This fault is an imprecise abort but the bit is only valid when a precise abort occurs.
reserved	29:19	sro,ns sraz,n snsro	0x0	read undefined
data_read_err	18	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs a data read: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
data_write_err	17	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the BRESP bus, after the DMA channel thread performs a data write: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
instr_fetch_err	16	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is a precise abort.
reserved	15:14	sro,ns sraz,n snsro	0x0	read undefined

Field Name	Bits	Type	Reset Value	Description
st_data_unavailable	13	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO did not contain the data to enable the DMAC to perform the DMAST: 0 = MFIFO contains all the data to enable the DMAST to complete 1 = previous DMALDs have not put enough data in the MFIFO to enable the DMAST to complete. This fault is a precise abort.
mfifo_err	12	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO prevented the DMA channel thread from executing DMALD or DMAST. Depending on the instruction: DMALD 0 = MFIFO contains sufficient space 1 = MFIFO is too small to hold the data that DMALD requires. DMAST 0 = MFIFO contains sufficient data 1 = MFIFO is too small to store the data to enable DMAST to complete. This fault is an imprecise abort.
reserved	11:8	sro,ns sraz,n snsro	0x0	read undefined
ch_rdwr_err	7	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to program the CCRn Register to perform a secure read or secure write: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to perform a secure read or secure write. This fault is a precise abort.
ch_periph_err	6	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to execute DMAWFP, DMALDP, DMASTP, or DMAFLUSHP with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFP to wait for a secure peripheral DMALDP or DMASTP to notify a secure peripheral DMAFLUSHP to flush a secure peripheral. This fault is a precise abort.

Field Name	Bits	Type	Reset Value	Description
ch_evnt_err	5	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread attempts to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFE to wait for a secure event DMASEV to create a secure event or secure interrupt. This fault is a precise abort.
reserved	4:2	sro,ns sraz,n snsro	0x0	read undefined
operand_invalid	1	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand. This fault is a precise abort.
undef_instr	0	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction. This fault is a precise abort.

Register ([dmac](#)) FTR1

Name	FTR1
Software Name	XDmaPs_FTCn_OFFSET(1)
Relative Address	0x00000044
Absolute Address	dmac0_ns: 0xF8004044 dmac0_s: 0xF8003044
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Default type for DMA channel 1

Register FTR1 Details

Field Name	Bits	Type	Reset Value	Description
lockup_err	31	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel has locked-up because of resource starvation: 0 = DMA channel has adequate resources 1 = DMA channel has locked-up because of insufficient resources. This fault is an imprecise abort.
dbg_instr	30	sro,ns sraz,n snsro	0x0	If the DMA channel aborts, this bit indicates whether the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface. This fault is an imprecise abort but the bit is only valid when a precise abort occurs.
reserved	29:19	sro,ns sraz,n snsro	0x0	read undefined
data_read_err	18	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs a data read: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
data_write_err	17	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the BRESP bus, after the DMA channel thread performs a data write: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
instr_fetch_err	16	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is a precise abort.
reserved	15:14	sro,ns sraz,n snsro	0x0	read undefined

Field Name	Bits	Type	Reset Value	Description
st_data_unavailable	13	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO did not contain the data to enable the DMAC to perform the DMAST: 0 = MFIFO contains all the data to enable the DMAST to complete 1 = previous DMALDs have not put enough data in the MFIFO to enable the DMAST to complete. This fault is a precise abort.
mfifo_err	12	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO prevented the DMA channel thread from executing DMALD or DMAST. Depending on the instruction: DMALD 0 = MFIFO contains sufficient space 1 = MFIFO is too small to hold the data that DMALD requires. DMAST 0 = MFIFO contains sufficient data 1 = MFIFO is too small to store the data to enable DMAST to complete. This fault is an imprecise abort.
reserved	11:8	sro,ns sraz,n snsro	0x0	read undefined
ch_rdwr_err	7	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to program the CCRn Register to perform a secure read or secure write: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to perform a secure read or secure write. This fault is a precise abort.
ch_periph_err	6	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to execute DMAWFP, DMALDP, DMASTP, or DMAFLUSHP with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFP to wait for a secure peripheral DMALDP or DMASTP to notify a secure peripheral DMAFLUSHP to flush a secure peripheral. This fault is a precise abort.

Field Name	Bits	Type	Reset Value	Description
ch_evnt_err	5	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread attempts to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFE to wait for a secure event DMASEV to create a secure event or secure interrupt. This fault is a precise abort.
reserved	4:2	sro,ns sraz,n snsro	0x0	read undefined
operand_invalid	1	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand. This fault is a precise abort.
undef_instr	0	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction. This fault is a precise abort.

Register ([dmac](#)) FTR2

Name	FTR2
Software Name	XDmaPs_FTCn_OFFSET(2)
Relative Address	0x00000048
Absolute Address	dmac0_ns: 0xF8004048 dmac0_s: 0xF8003048
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Default type for DMA channel 2

Register FTR2 Details

Field Name	Bits	Type	Reset Value	Description
lockup_err	31	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel has locked-up because of resource starvation: 0 = DMA channel has adequate resources 1 = DMA channel has locked-up because of insufficient resources. This fault is an imprecise abort.
dbg_instr	30	sro,ns sraz,n snsro	0x0	If the DMA channel aborts, this bit indicates whether the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface. This fault is an imprecise abort but the bit is only valid when a precise abort occurs.
reserved	29:19	sro,ns sraz,n snsro	0x0	read undefined
data_read_err	18	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs a data read: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
data_write_err	17	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the BRESP bus, after the DMA channel thread performs a data write: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
instr_fetch_err	16	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is a precise abort.
reserved	15:14	sro,ns sraz,n snsro	0x0	read undefined

Field Name	Bits	Type	Reset Value	Description
st_data_unavailable	13	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO did not contain the data to enable the DMAC to perform the DMAST: 0 = MFIFO contains all the data to enable the DMAST to complete 1 = previous DMALDs have not put enough data in the MFIFO to enable the DMAST to complete. This fault is a precise abort.
mfifo_err	12	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO prevented the DMA channel thread from executing DMALD or DMAST. Depending on the instruction: DMALD 0 = MFIFO contains sufficient space 1 = MFIFO is too small to hold the data that DMALD requires. DMAST 0 = MFIFO contains sufficient data 1 = MFIFO is too small to store the data to enable DMAST to complete. This fault is an imprecise abort.
reserved	11:8	sro,ns sraz,n snsro	0x0	read undefined
ch_rdwr_err	7	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to program the CCRn Register to perform a secure read or secure write: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to perform a secure read or secure write. This fault is a precise abort.
ch_periph_err	6	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to execute DMAWFP, DMALDP, DMASTP, or DMAFLUSHP with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFP to wait for a secure peripheral DMALDP or DMASTP to notify a secure peripheral DMAFLUSHP to flush a secure peripheral. This fault is a precise abort.

Field Name	Bits	Type	Reset Value	Description
ch_evnt_err	5	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread attempts to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFE to wait for a secure event DMASEV to create a secure event or secure interrupt. This fault is a precise abort.
reserved	4:2	sro,ns sraz,n snsro	0x0	read undefined
operand_invalid	1	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand. This fault is a precise abort.
undef_instr	0	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction. This fault is a precise abort.

Register ([dmac](#)) FTR3

Name	FTR3
Software Name	XDmaPs_FTCn_OFFSET(3)
Relative Address	0x0000004C
Absolute Address	dmac0_ns: 0xF800404C dmac0_s: 0xF800304C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Default type for DMA channel 3

Register FTR3 Details

Field Name	Bits	Type	Reset Value	Description
lockup_err	31	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel has locked-up because of resource starvation: 0 = DMA channel has adequate resources 1 = DMA channel has locked-up because of insufficient resources. This fault is an imprecise abort.
dbg_instr	30	sro,ns sraz,n snsro	0x0	If the DMA channel aborts, this bit indicates whether the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface. This fault is an imprecise abort but the bit is only valid when a precise abort occurs.
reserved	29:19	sro,ns sraz,n snsro	0x0	read undefined
data_read_err	18	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs a data read: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
data_write_err	17	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the BRESP bus, after the DMA channel thread performs a data write: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
instr_fetch_err	16	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is a precise abort.
reserved	15:14	sro,ns sraz,n snsro	0x0	read undefined

Field Name	Bits	Type	Reset Value	Description
st_data_unavailable	13	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO did not contain the data to enable the DMAC to perform the DMAST: 0 = MFIFO contains all the data to enable the DMAST to complete 1 = previous DMALDs have not put enough data in the MFIFO to enable the DMAST to complete. This fault is a precise abort.
mfifo_err	12	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO prevented the DMA channel thread from executing DMALD or DMAST. Depending on the instruction: DMALD 0 = MFIFO contains sufficient space 1 = MFIFO is too small to hold the data that DMALD requires. DMAST 0 = MFIFO contains sufficient data 1 = MFIFO is too small to store the data to enable DMAST to complete. This fault is an imprecise abort.
reserved	11:8	sro,ns sraz,n snsro	0x0	read undefined
ch_rdwr_err	7	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to program the CCRn Register to perform a secure read or secure write: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to perform a secure read or secure write. This fault is a precise abort.
ch_periph_err	6	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to execute DMAWFP, DMALDP, DMASTP, or DMAFLUSHP with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFP to wait for a secure peripheral DMALDP or DMASTP to notify a secure peripheral DMAFLUSHP to flush a secure peripheral. This fault is a precise abort.

Field Name	Bits	Type	Reset Value	Description
ch_evnt_err	5	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread attempts to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFE to wait for a secure event DMASEV to create a secure event or secure interrupt. This fault is a precise abort.
reserved	4:2	sro,ns sraz,n snsro	0x0	read undefined
operand_invalid	1	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand. This fault is a precise abort.
undef_instr	0	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction. This fault is a precise abort.

Register ([dmac](#)) FTR4

Name	FTR4
Software Name	XDmaPs_FTCn_OFFSET(4)
Relative Address	0x00000050
Absolute Address	dmac0_ns: 0xF8004050 dmac0_s: 0xF8003050
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Default type for DMA channel 4

Register FTR4 Details

Field Name	Bits	Type	Reset Value	Description
lockup_err	31	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel has locked-up because of resource starvation: 0 = DMA channel has adequate resources 1 = DMA channel has locked-up because of insufficient resources. This fault is an imprecise abort.
dbg_instr	30	sro,ns sraz,n snsro	0x0	If the DMA channel aborts, this bit indicates whether the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface. This fault is an imprecise abort but the bit is only valid when a precise abort occurs.
reserved	29:19	sro,ns sraz,n snsro	0x0	read undefined
data_read_err	18	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs a data read: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
data_write_err	17	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the BRESP bus, after the DMA channel thread performs a data write: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
instr_fetch_err	16	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is a precise abort.
reserved	15:14	sro,ns sraz,n snsro	0x0	read undefined

Field Name	Bits	Type	Reset Value	Description
st_data_unavailable	13	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO did not contain the data to enable the DMAC to perform the DMAST: 0 = MFIFO contains all the data to enable the DMAST to complete 1 = previous DMALDs have not put enough data in the MFIFO to enable the DMAST to complete. This fault is a precise abort.
mfifo_err	12	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO prevented the DMA channel thread from executing DMALD or DMAST. Depending on the instruction: DMALD 0 = MFIFO contains sufficient space 1 = MFIFO is too small to hold the data that DMALD requires. DMAST 0 = MFIFO contains sufficient data 1 = MFIFO is too small to store the data to enable DMAST to complete. This fault is an imprecise abort.
reserved	11:8	sro,ns sraz,n snsro	0x0	read undefined
ch_rdwr_err	7	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to program the CCRn Register to perform a secure read or secure write: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to perform a secure read or secure write. This fault is a precise abort.
ch_periph_err	6	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to execute DMAWFP, DMALDP, DMASTP, or DMAFLUSHP with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFP to wait for a secure peripheral DMALDP or DMASTP to notify a secure peripheral DMAFLUSHP to flush a secure peripheral. This fault is a precise abort.

Field Name	Bits	Type	Reset Value	Description
ch_evnt_err	5	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread attempts to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFE to wait for a secure event DMASEV to create a secure event or secure interrupt. This fault is a precise abort.
reserved	4:2	sro,ns sraz,n snsro	0x0	read undefined
operand_invalid	1	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand. This fault is a precise abort.
undef_instr	0	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction. This fault is a precise abort.

Register ([dmac](#)) FTR5

Name	FTR5
Software Name	XDmaPs_FTCn_OFFSET(5)
Relative Address	0x00000054
Absolute Address	dmac0_ns: 0xF8004054 dmac0_s: 0xF8003054
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Default type for DMA channel 5

Register FTR5 Details

Field Name	Bits	Type	Reset Value	Description
lockup_err	31	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel has locked-up because of resource starvation: 0 = DMA channel has adequate resources 1 = DMA channel has locked-up because of insufficient resources. This fault is an imprecise abort.
dbg_instr	30	sro,ns sraz,n snsro	0x0	If the DMA channel aborts, this bit indicates whether the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface. This fault is an imprecise abort but the bit is only valid when a precise abort occurs.
reserved	29:19	sro,ns sraz,n snsro	0x0	read undefined
data_read_err	18	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs a data read: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
data_write_err	17	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the BRESP bus, after the DMA channel thread performs a data write: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
instr_fetch_err	16	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is a precise abort.
reserved	15:14	sro,ns sraz,n snsro	0x0	read undefined

Field Name	Bits	Type	Reset Value	Description
st_data_unavailable	13	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO did not contain the data to enable the DMAC to perform the DMAST: 0 = MFIFO contains all the data to enable the DMAST to complete 1 = previous DMALDs have not put enough data in the MFIFO to enable the DMAST to complete. This fault is a precise abort.
mfifo_err	12	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO prevented the DMA channel thread from executing DMALD or DMAST. Depending on the instruction: DMALD 0 = MFIFO contains sufficient space 1 = MFIFO is too small to hold the data that DMALD requires. DMAST 0 = MFIFO contains sufficient data 1 = MFIFO is too small to store the data to enable DMAST to complete. This fault is an imprecise abort.
reserved	11:8	sro,ns sraz,n snsro	0x0	read undefined
ch_rdwr_err	7	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to program the CCRn Register to perform a secure read or secure write: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to perform a secure read or secure write. This fault is a precise abort.
ch_periph_err	6	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to execute DMAWFP, DMALDP, DMASTP, or DMAFLUSHP with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFP to wait for a secure peripheral DMALDP or DMASTP to notify a secure peripheral DMAFLUSHP to flush a secure peripheral. This fault is a precise abort.

Field Name	Bits	Type	Reset Value	Description
ch_evnt_err	5	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread attempts to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFE to wait for a secure event DMASEV to create a secure event or secure interrupt. This fault is a precise abort.
reserved	4:2	sro,ns sraz,n snsro	0x0	read undefined
operand_invalid	1	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand. This fault is a precise abort.
undef_instr	0	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction. This fault is a precise abort.

Register ([dmac](#)) FTR6

Name	FTR6
Software Name	XDmaPs_FTCn_OFFSET(6)
Relative Address	0x00000058
Absolute Address	dmac0_ns: 0xF8004058 dmac0_s: 0xF8003058
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Default type for DMA channel 6

Register FTR6 Details

Field Name	Bits	Type	Reset Value	Description
lockup_err	31	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel has locked-up because of resource starvation: 0 = DMA channel has adequate resources 1 = DMA channel has locked-up because of insufficient resources. This fault is an imprecise abort.
dbg_instr	30	sro,ns sraz,n snsro	0x0	If the DMA channel aborts, this bit indicates whether the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface. This fault is an imprecise abort but the bit is only valid when a precise abort occurs.
reserved	29:19	sro,ns sraz,n snsro	0x0	read undefined
data_read_err	18	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs a data read: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
data_write_err	17	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the BRESP bus, after the DMA channel thread performs a data write: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
instr_fetch_err	16	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is a precise abort.
reserved	15:14	sro,ns sraz,n snsro	0x0	read undefined

Field Name	Bits	Type	Reset Value	Description
st_data_unavailable	13	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO did not contain the data to enable the DMAC to perform the DMAST: 0 = MFIFO contains all the data to enable the DMAST to complete 1 = previous DMALDs have not put enough data in the MFIFO to enable the DMAST to complete. This fault is a precise abort.
mfifo_err	12	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO prevented the DMA channel thread from executing DMALD or DMAST. Depending on the instruction: DMALD 0 = MFIFO contains sufficient space 1 = MFIFO is too small to hold the data that DMALD requires. DMAST 0 = MFIFO contains sufficient data 1 = MFIFO is too small to store the data to enable DMAST to complete. This fault is an imprecise abort.
reserved	11:8	sro,ns sraz,n snsro	0x0	read undefined
ch_rdwr_err	7	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to program the CCRn Register to perform a secure read or secure write: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to perform a secure read or secure write. This fault is a precise abort.
ch_periph_err	6	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to execute DMAWFP, DMALDP, DMASTP, or DMAFLUSHP with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFP to wait for a secure peripheral DMALDP or DMASTP to notify a secure peripheral DMAFLUSHP to flush a secure peripheral. This fault is a precise abort.

Field Name	Bits	Type	Reset Value	Description
ch_evnt_err	5	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread attempts to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFE to wait for a secure event DMASEV to create a secure event or secure interrupt. This fault is a precise abort.
reserved	4:2	sro,ns sraz,n snsro	0x0	read undefined
operand_invalid	1	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand. This fault is a precise abort.
undef_instr	0	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction. This fault is a precise abort.

Register ([dmac](#)) FTR7

Name	FTR7
Software Name	XDmaPs_FTCn_OFFSET(7)
Relative Address	0x0000005C
Absolute Address	dmac0_ns: 0xF800405C dmac0_s: 0xF800305C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Default type for DMA channel 7

Register FTR7 Details

Field Name	Bits	Type	Reset Value	Description
lockup_err	31	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel has locked-up because of resource starvation: 0 = DMA channel has adequate resources 1 = DMA channel has locked-up because of insufficient resources. This fault is an imprecise abort.
dbg_instr	30	sro,ns sraz,n snsro	0x0	If the DMA channel aborts, this bit indicates whether the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface. This fault is an imprecise abort but the bit is only valid when a precise abort occurs.
reserved	29:19	sro,ns sraz,n snsro	0x0	read undefined
data_read_err	18	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs a data read: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
data_write_err	17	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the BRESP bus, after the DMA channel thread performs a data write: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
instr_fetch_err	16	sro,ns sraz,n snsro	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is a precise abort.
reserved	15:14	sro,ns sraz,n snsro	0x0	read undefined

Field Name	Bits	Type	Reset Value	Description
st_data_unavailable	13	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO did not contain the data to enable the DMAC to perform the DMAST: 0 = MFIFO contains all the data to enable the DMAST to complete 1 = previous DMALDs have not put enough data in the MFIFO to enable the DMAST to complete. This fault is a precise abort.
mfifo_err	12	sro,ns sraz,n snsro	0x0	Indicates whether the MFIFO prevented the DMA channel thread from executing DMALD or DMAST. Depending on the instruction: DMALD 0 = MFIFO contains sufficient space 1 = MFIFO is too small to hold the data that DMALD requires. DMAST 0 = MFIFO contains sufficient data 1 = MFIFO is too small to store the data to enable DMAST to complete. This fault is an imprecise abort.
reserved	11:8	sro,ns sraz,n snsro	0x0	read undefined
ch_rdwr_err	7	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to program the CCRn Register to perform a secure read or secure write: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to perform a secure read or secure write. This fault is a precise abort.
ch_periph_err	6	sro,ns sraz,n snsro	0x0	Indicates whether a DMA channel thread, in the Non-secure state, attempts to execute DMAWFP, DMALDP, DMASTP, or DMAFLUSHP with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFP to wait for a secure peripheral DMALDP or DMASTP to notify a secure peripheral DMAFLUSHP to flush a secure peripheral. This fault is a precise abort.

Field Name	Bits	Type	Reset Value	Description
ch_evnt_err	5	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread attempts to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: DMAWFE to wait for a secure event DMASEV to create a secure event or secure interrupt. This fault is a precise abort.
reserved	4:2	sro,ns sraz,n snsro	0x0	read undefined
operand_invalid	1	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand. This fault is a precise abort.
undef_instr	0	sro,ns sraz,n snsro	0x0	Indicates whether the DMA channel thread was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction. This fault is a precise abort.

Register ([dmac](#)) CSR0

Name	CSR0
Software Name	CS0
Relative Address	0x00000100
Absolute Address	dmac0_ns: 0xF8004100 dmac0_s: 0xF8003100
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Channel status for DMA channel 0

Register CSR0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rud	0x0	reserved, read undefined
CNS	21	sro,ns sraz,n snsro	0x0	The channel non-secure bit provides the security of the DMA channel: 0 = DMA channel operates in the Secure state 1 = DMA channel operates in the Non-secure state.
reserved	20:16	rud	0x0	reserved, read undefined
dmawfp_periph	15	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the periph operand was set: 0 = DMAWFP executed with the periph operand not set 1 = DMAWFP executed with the periph operand set.
dmawfp_b_ns	14	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the burst or single operand were set: 0 = DMAWFP executed with the single operand set 1 = DMAWFP executed with the burst operand set
reserved	13:9	rud	0x0	reserved, read undefined

Field Name	Bits	Type	Reset Value	Description
wakeup_num	8:4	sro,ns sraz,n snsro	0x0	<p>If the DMA channel is in the Waiting for event state, or the Waiting for peripheral state, then these bits</p> <p>indicate the event or peripheral number that the channel is waiting for:</p> <p>b00000 = DMA channel is waiting for event, or peripheral, 0</p> <p>b00001 = DMA channel is waiting for event, or peripheral, 1</p> <p>b00010 = DMA channel is waiting for event, or peripheral, 2</p> <p>.</p> <p>.</p> <p>.</p> <p>b11111 = DMA channel is waiting for event, or peripheral, 31.</p>
channel_status	3:0	sro,ns sraz,n snsro	0x0	<p>The channel status encoding is:</p> <p>b0000 = Stopped</p> <p>b0001 = Executing</p> <p>b0010 = Cache miss</p> <p>b0011 = Updating PC</p> <p>b0100 = Waiting for event</p> <p>b0101 = At barrier</p> <p>b0110 = reserved</p> <p>b0111 = Waiting for peripheral</p> <p>b1000 = Killing</p> <p>b1001 = Completing</p> <p>b1010-b1101 = reserved</p> <p>b1110 = Faulting completing</p> <p>b1111 = Faulting.</p>

Register ([dmac](#)) CPC0

Name	CPC0
Relative Address	0x00000104
Absolute Address	dmac0_ns: 0xF8004104 dmac0_s: 0xF8003104
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Channel PC for DMA channel 0

Register CPC0 Details

Field Name	Bits	Type	Reset Value	Description
pc_chnl	31:0	sro,ns sraz,n snsro	0x0	Program counter for the DMA channel n thread, where n depends on the address of the register

Register ([dmac](#)) CSR1

Name	CSR1
Software Name	XDmaPs_CSn_OFFSET(1)
Relative Address	0x00000108
Absolute Address	dmac0_ns: 0xF8004108 dmac0_s: 0xF8003108
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Channel status for DMA channel 1

Register CSR1 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rud	0x0	reserved, read undefined
CNS	21	sro,ns sraz,n snsro	0x0	The channel non-secure bit provides the security of the DMA channel: 0 = DMA channel operates in the Secure state 1 = DMA channel operates in the Non-secure state.
reserved	20:16	rud	0x0	reserved, read undefined
dmawfp_periph	15	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the periph operand was set: 0 = DMAWFP executed with the periph operand not set 1 = DMAWFP executed with the periph operand set.
dmawfp_b_ns	14	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the burst or single operand were set: 0 = DMAWFP executed with the single operand set 1 = DMAWFP executed with the burst operand set

Field Name	Bits	Type	Reset Value	Description
reserved	13:9	rud	0x0	reserved, read undefined
wakeup_num	8:4	sro,ns sraz,n snsro	0x0	<p>If the DMA channel is in the Waiting for event state, or the Waiting for peripheral state, then these bits</p> <p>indicate the event or peripheral number that the channel is waiting for:</p> <p>b00000 = DMA channel is waiting for event, or peripheral, 0</p> <p>b00001 = DMA channel is waiting for event, or peripheral, 1</p> <p>b00010 = DMA channel is waiting for event, or peripheral, 2</p> <p>.</p> <p>.</p> <p>.</p> <p>b11111 = DMA channel is waiting for event, or peripheral, 31.</p>
channel_status	3:0	sro,ns sraz,n snsro	0x0	<p>The channel status encoding is:</p> <p>b0000 = Stopped</p> <p>b0001 = Executing</p> <p>b0010 = Cache miss</p> <p>b0011 = Updating PC</p> <p>b0100 = Waiting for event</p> <p>b0101 = At barrier</p> <p>b0110 = reserved</p> <p>b0111 = Waiting for peripheral</p> <p>b1000 = Killing</p> <p>b1001 = Completing</p> <p>b1010-b1101 = reserved</p> <p>b1110 = Faulting completing</p> <p>b1111 = Faulting.</p>

Register ([dmac](#)) CPC1

Name	CPC1
Software Name	XDmaPs_CPCn_OFFSET(1)
Relative Address	0x0000010C
Absolute Address	dmac0_ns: 0xF800410C dmac0_s: 0xF800310C
Width	32 bits
Access Type	mixed

Reset Value 0x00000000

Description Channel PC for DMA channel 1

Register CPC1 Details

Field Name	Bits	Type	Reset Value	Description
pc_chnl	31:0	sro,ns sraz,n snsro	0x0	Program counter for the DMA channel n thread, where n depends on the address of the register

Register ([dmac](#)) CSR2

Name CSR2

Software Name XDmaPs_CSn_OFFSET(2)

Relative Address 0x00000110

Absolute Address dmac0_ns: 0xF8004110
dmac0_s: 0xF8003110

Width 32 bits

Access Type mixed

Reset Value 0x00000000

Description Channel status for DMA channel 2

Register CSR2 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rud	0x0	reserved, read undefined
CNS	21	sro,ns sraz,n snsro	0x0	The channel non-secure bit provides the security of the DMA channel: 0 = DMA channel operates in the Secure state 1 = DMA channel operates in the Non-secure state.
reserved	20:16	rud	0x0	reserved, read undefined
dmawfp_periph	15	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the periph operand was set: 0 = DMAWFP executed with the periph operand not set 1 = DMAWFP executed with the periph operand set.

Field Name	Bits	Type	Reset Value	Description
dmawfp_b_ns	14	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the burst or single operand were set: 0 = DMAWFP executed with the single operand set 1 = DMAWFP executed with the burst operand set
reserved	13:9	rud	0x0	reserved, read undefined
wakeup_num	8:4	sro,ns sraz,n snsro	0x0	If the DMA channel is in the Waiting for event state, or the Waiting for peripheral state, then these bits indicate the event or peripheral number that the channel is waiting for: b00000 = DMA channel is waiting for event, or peripheral, 0 b00001 = DMA channel is waiting for event, or peripheral, 1 b00010 = DMA channel is waiting for event, or peripheral, 2 . . . b11111 = DMA channel is waiting for event, or peripheral, 31.
channel_status	3:0	sro,ns sraz,n snsro	0x0	The channel status encoding is: b0000 = Stopped b0001 = Executing b0010 = Cache miss b0011 = Updating PC b0100 = Waiting for event b0101 = At barrier b0110 = reserved b0111 = Waiting for peripheral b1000 = Killing b1001 = Completing b1010-b1101 = reserved b1110 = Faulting completing b1111 = Faulting.

Register ([dmac](#)) CPC2

Name CPC2
Software Name XDmaPs_CPCn_OFFSET(2)

Relative Address	0x00000114
Absolute Address	dmac0_ns: 0xF8004114 dmac0_s: 0xF8003114
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Channel PC for DMA channel 2

Register CPC2 Details

Field Name	Bits	Type	Reset Value	Description
pc_chnl	31:0	sro,ns sraz,n snsro	0x0	Program counter for the DMA channel n thread, where n depends on the address of the register

Register ([dmac](#)) CSR3

Name	CSR3
Software Name	XDmaPs_CSn_OFFSET(3)
Relative Address	0x00000118
Absolute Address	dmac0_ns: 0xF8004118 dmac0_s: 0xF8003118
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Channel status for DMA channel 3

Register CSR3 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rud	0x0	reserved, read undefined
CNS	21	sro,ns sraz,n snsro	0x0	The channel non-secure bit provides the security of the DMA channel: 0 = DMA channel operates in the Secure state 1 = DMA channel operates in the Non-secure state.
reserved	20:16	rud	0x0	reserved, read undefined

Field Name	Bits	Type	Reset Value	Description
dmawfp_periph	15	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the periph operand was set: 0 = DMAWFP executed with the periph operand not set 1 = DMAWFP executed with the periph operand set.
dmawfp_b_ns	14	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the burst or single operand were set: 0 = DMAWFP executed with the single operand set 1 = DMAWFP executed with the burst operand set
reserved	13:9	rud	0x0	reserved, read undefined

Field Name	Bits	Type	Reset Value	Description
wakeup_num	8:4	sro,ns sraz,n snsro	0x0	<p>If the DMA channel is in the Waiting for event state, or the Waiting for peripheral state, then these bits</p> <p>indicate the event or peripheral number that the channel is waiting for:</p> <p>b00000 = DMA channel is waiting for event, or peripheral, 0</p> <p>b00001 = DMA channel is waiting for event, or peripheral, 1</p> <p>b00010 = DMA channel is waiting for event, or peripheral, 2</p> <p>.</p> <p>.</p> <p>.</p> <p>b11111 = DMA channel is waiting for event, or peripheral, 31.</p>
channel_status	3:0	sro,ns sraz,n snsro	0x0	<p>The channel status encoding is:</p> <p>b0000 = Stopped</p> <p>b0001 = Executing</p> <p>b0010 = Cache miss</p> <p>b0011 = Updating PC</p> <p>b0100 = Waiting for event</p> <p>b0101 = At barrier</p> <p>b0110 = reserved</p> <p>b0111 = Waiting for peripheral</p> <p>b1000 = Killing</p> <p>b1001 = Completing</p> <p>b1010-b1101 = reserved</p> <p>b1110 = Faulting completing</p> <p>b1111 = Faulting.</p>

Register ([dmac](#)) CPC3

Name	CPC3
Software Name	XDmaPs_CPCn_OFFSET(3)
Relative Address	0x0000011C
Absolute Address	dmac0_ns: 0xF800411C dmac0_s: 0xF800311C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000

Description Channel PC for DMA channel 3

Register CPC3 Details

Field Name	Bits	Type	Reset Value	Description
pc_chnl	31:0	sro,ns sraz,n snsro	0x0	Program counter for the DMA channel n thread, where n depends on the address of the register

Register ([dmac](#)) CSR4

Name CSR4

Software Name XDmaPs_CS_n_OFFSET(4)

Relative Address 0x00000120

Absolute Address dmac0_ns: 0xF8004120
dmac0_s: 0xF8003120

Width 32 bits

Access Type mixed

Reset Value 0x00000000

Description Channel status for DMA channel 4

Register CSR4 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rud	0x0	reserved, read undefined
CNS	21	sro,ns sraz,n snsro	0x0	The channel non-secure bit provides the security of the DMA channel: 0 = DMA channel operates in the Secure state 1 = DMA channel operates in the Non-secure state.
reserved	20:16	rud	0x0	reserved, read undefined
dmawfp_periph	15	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the periph operand was set: 0 = DMAWFP executed with the periph operand not set 1 = DMAWFP executed with the periph operand set.

Field Name	Bits	Type	Reset Value	Description
dmawfp_b_ns	14	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the burst or single operand were set: 0 = DMAWFP executed with the single operand set 1 = DMAWFP executed with the burst operand set
reserved	13:9	rud	0x0	reserved, read undefined
wakeup_num	8:4	sro,ns sraz,n snsro	0x0	If the DMA channel is in the Waiting for event state, or the Waiting for peripheral state, then these bits indicate the event or peripheral number that the channel is waiting for: b00000 = DMA channel is waiting for event, or peripheral, 0 b00001 = DMA channel is waiting for event, or peripheral, 1 b00010 = DMA channel is waiting for event, or peripheral, 2 . . . b11111 = DMA channel is waiting for event, or peripheral, 31.
channel_status	3:0	sro,ns sraz,n snsro	0x0	The channel status encoding is: b0000 = Stopped b0001 = Executing b0010 = Cache miss b0011 = Updating PC b0100 = Waiting for event b0101 = At barrier b0110 = reserved b0111 = Waiting for peripheral b1000 = Killing b1001 = Completing b1010-b1101 = reserved b1110 = Faulting completing b1111 = Faulting.

Register ([dmac](#)) CPC4

Name CPC4
Software Name XDmaPs_CPCn_OFFSET(4)

Relative Address	0x00000124
Absolute Address	dmac0_ns: 0xF8004124 dmac0_s: 0xF8003124
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Channel PC for DMA channel 4

Register CPC4 Details

Field Name	Bits	Type	Reset Value	Description
pc_chnl	31:0	sro,ns sraz,n snsro	0x0	Program counter for the DMA channel n thread, where n depends on the address of the register

Register ([dmac](#)) CSR5

Name	CSR5
Software Name	XDmaPs_CSn_OFFSET(5)
Relative Address	0x00000128
Absolute Address	dmac0_ns: 0xF8004128 dmac0_s: 0xF8003128
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Channel status for DMA channel 5

Register CSR5 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rud	0x0	reserved, read undefined
CNS	21	sro,ns sraz,n snsro	0x0	The channel non-secure bit provides the security of the DMA channel: 0 = DMA channel operates in the Secure state 1 = DMA channel operates in the Non-secure state.
reserved	20:16	rud	0x0	reserved, read undefined

Field Name	Bits	Type	Reset Value	Description
dmawfp_periph	15	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the periph operand was set: 0 = DMAWFP executed with the periph operand not set 1 = DMAWFP executed with the periph operand set.
dmawfp_b_ns	14	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the burst or single operand were set: 0 = DMAWFP executed with the single operand set 1 = DMAWFP executed with the burst operand set
reserved	13:9	rud	0x0	reserved, read undefined

Field Name	Bits	Type	Reset Value	Description
wakeup_num	8:4	sro,ns sraz,n snsro	0x0	<p>If the DMA channel is in the Waiting for event state, or the Waiting for peripheral state, then these bits</p> <p>indicate the event or peripheral number that the channel is waiting for:</p> <p>b00000 = DMA channel is waiting for event, or peripheral, 0</p> <p>b00001 = DMA channel is waiting for event, or peripheral, 1</p> <p>b00010 = DMA channel is waiting for event, or peripheral, 2</p> <p>.</p> <p>.</p> <p>.</p> <p>b11111 = DMA channel is waiting for event, or peripheral, 31.</p>
channel_status	3:0	sro,ns sraz,n snsro	0x0	<p>The channel status encoding is:</p> <p>b0000 = Stopped</p> <p>b0001 = Executing</p> <p>b0010 = Cache miss</p> <p>b0011 = Updating PC</p> <p>b0100 = Waiting for event</p> <p>b0101 = At barrier</p> <p>b0110 = reserved</p> <p>b0111 = Waiting for peripheral</p> <p>b1000 = Killing</p> <p>b1001 = Completing</p> <p>b1010-b1101 = reserved</p> <p>b1110 = Faulting completing</p> <p>b1111 = Faulting.</p>

Register ([dmac](#)) CPC5

Name	CPC5
Software Name	XDmaPs_CPCn_OFFSET(5)
Relative Address	0x0000012C
Absolute Address	dmac0_ns: 0xF800412C dmac0_s: 0xF800312C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000

Description Channel PC for DMA channel 5

Register CPC5 Details

Field Name	Bits	Type	Reset Value	Description
pc_chnl	31:0	sro,ns sraz,n snsro	0x0	Program counter for the DMA channel n thread, where n depends on the address of the register

Register ([dmac](#)) CSR6

Name CSR6

Software Name XDmaPs_CSn_OFFSET(6)

Relative Address 0x00000130

Absolute Address dmac0_ns: 0xF8004130
dmac0_s: 0xF8003130

Width 32 bits

Access Type mixed

Reset Value 0x00000000

Description Channel status for DMA channel 6

Register CSR6 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rud	0x0	reserved, read undefined
CNS	21	sro,ns sraz,n snsro	0x0	The channel non-secure bit provides the security of the DMA channel: 0 = DMA channel operates in the Secure state 1 = DMA channel operates in the Non-secure state.
reserved	20:16	rud	0x0	reserved, read undefined
dmawfp_periph	15	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the periph operand was set: 0 = DMAWFP executed with the periph operand not set 1 = DMAWFP executed with the periph operand set.

Field Name	Bits	Type	Reset Value	Description
dmawfp_b_ns	14	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the burst or single operand were set: 0 = DMAWFP executed with the single operand set 1 = DMAWFP executed with the burst operand set
reserved	13:9	rud	0x0	reserved, read undefined
wakeup_num	8:4	sro,ns sraz,n snsro	0x0	If the DMA channel is in the Waiting for event state, or the Waiting for peripheral state, then these bits indicate the event or peripheral number that the channel is waiting for: b00000 = DMA channel is waiting for event, or peripheral, 0 b00001 = DMA channel is waiting for event, or peripheral, 1 b00010 = DMA channel is waiting for event, or peripheral, 2 . . . b11111 = DMA channel is waiting for event, or peripheral, 31.
channel_status	3:0	sro,ns sraz,n snsro	0x0	The channel status encoding is: b0000 = Stopped b0001 = Executing b0010 = Cache miss b0011 = Updating PC b0100 = Waiting for event b0101 = At barrier b0110 = reserved b0111 = Waiting for peripheral b1000 = Killing b1001 = Completing b1010-b1101 = reserved b1110 = Faulting completing b1111 = Faulting.

Register ([dmac](#)) CPC6

Name CPC6
Software Name XDmaPs_CPCn_OFFSET(6)

Relative Address	0x00000134
Absolute Address	dmac0_ns: 0xF8004134 dmac0_s: 0xF8003134
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Channel PC for DMA channel 6

Register CPC6 Details

Field Name	Bits	Type	Reset Value	Description
pc_chnl	31:0	sro,ns sraz,n snsro	0x0	Program counter for the DMA channel n thread, where n depends on the address of the register

Register ([dmac](#)) CSR7

Name	CSR7
Software Name	XDmaPs_CSn_OFFSET(7)
Relative Address	0x00000138
Absolute Address	dmac0_ns: 0xF8004138 dmac0_s: 0xF8003138
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Channel status for DMA channel 7

Register CSR7 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rud	0x0	reserved, read undefined
CNS	21	sro,ns sraz,n snsro	0x0	The channel non-secure bit provides the security of the DMA channel: 0 = DMA channel operates in the Secure state 1 = DMA channel operates in the Non-secure state.
reserved	20:16	rud	0x0	reserved, read undefined

Field Name	Bits	Type	Reset Value	Description
dmawfp_periph	15	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the periph operand was set: 0 = DMAWFP executed with the periph operand not set 1 = DMAWFP executed with the periph operand set.
dmawfp_b_ns	14	sro,ns sraz,n snsro	0x0	When the DMA channel thread executes DMAWFP, this bit indicates whether the burst or single operand were set: 0 = DMAWFP executed with the single operand set 1 = DMAWFP executed with the burst operand set
reserved	13:9	rud	0x0	reserved, read undefined

Field Name	Bits	Type	Reset Value	Description
wakeup_num	8:4	sro,ns sraz,n snsro	0x0	<p>If the DMA channel is in the Waiting for event state, or the Waiting for peripheral state, then these bits</p> <p>indicate the event or peripheral number that the channel is waiting for:</p> <p>b00000 = DMA channel is waiting for event, or peripheral, 0</p> <p>b00001 = DMA channel is waiting for event, or peripheral, 1</p> <p>b00010 = DMA channel is waiting for event, or peripheral, 2</p> <p>.</p> <p>.</p> <p>.</p> <p>b11111 = DMA channel is waiting for event, or peripheral, 31.</p>
channel_status	3:0	sro,ns sraz,n snsro	0x0	<p>The channel status encoding is:</p> <p>b0000 = Stopped</p> <p>b0001 = Executing</p> <p>b0010 = Cache miss</p> <p>b0011 = Updating PC</p> <p>b0100 = Waiting for event</p> <p>b0101 = At barrier</p> <p>b0110 = reserved</p> <p>b0111 = Waiting for peripheral</p> <p>b1000 = Killing</p> <p>b1001 = Completing</p> <p>b1010-b1101 = reserved</p> <p>b1110 = Faulting completing</p> <p>b1111 = Faulting.</p>

Register ([dmac](#)) CPC7

Name	CPC7
Software Name	XDmaPs_CPCn_OFFSET(7)
Relative Address	0x0000013C
Absolute Address	dmac0_ns: 0xF800413C dmac0_s: 0xF800313C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000

Description Channel PC for DMA channel 7

Register CPC7 Details

Field Name	Bits	Type	Reset Value	Description
pc_chnl	31:0	sro,ns sraz,n snsro	0x0	Program counter for the DMA channel n thread, where n depends on the address of the register

Register ([dmac](#)) SAR0

Name SAR0

Software Name SA_0

Relative Address 0x00000400

Absolute Address dmac0_ns: 0xF8004400
dmac0_s: 0xF8003400

Width 32 bits

Access Type mixed

Reset Value 0x00000000

Description source address for DMA channel 0

Register SAR0 Details

Field Name	Bits	Type	Reset Value	Description
src_addr	31:0	sro,ns sraz,n snsro	0x0	Address of the source data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) DAR0

Name DAR0

Software Name DA_0

Relative Address 0x00000404

Absolute Address dmac0_ns: 0xF8004404
dmac0_s: 0xF8003404

Width 32 bits

Access Type mixed

Reset Value 0x00000000

Description destination address for DMA channel 0

Register DAR0 Details

Field Name	Bits	Type	Reset Value	Description
dest_addr	31:0	sro,ns sraz,n snsro	0x0	Address for the destination data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) CCR0

Name	CCR0
Software Name	CC_0
Relative Address	0x00000408
Absolute Address	dmac0_ns: 0xF8004408 dmac0_s: 0xF8003408
Width	32 bits
Access Type	mixed
Reset Value	dmac0_ns: 0x00000000 dmac0_s: 0x00800200
Description	channel control for DMA channel 0

Register CCR0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	rud	0x0	reserved, read undefined
endian_swap_size	30:28	sro,ns sraz,n snsro	0x0	Table 3-22 on page 3-29 defines whether data can be swapped between little-endian (LE) and byte-invariant big-endian (BE-8) formats, and if so, also defines the natural width of the data independently of the source and destination transaction sizes. This enables unaligned data streams to use the full bus-width, and to be correctly transformed, irrespective of the source and destination address alignments. The format is identical to AxSIZE, except that b000 indicates that no swap must occur. Endian swap size Description b000 No swap, 8-bit data b001 Swap bytes within 16-bit data b010 Swap bytes within 32-bit data b011 Swap bytes within 64-bit data b100 Swap bytes within 128-bit data b101 Reserved b110 Reserved b111 Reserved

Field Name	Bits	Type	Reset Value	Description
dst_cache_ctrl	27:25	sro,ns sraz,n snsro	0x0	<p>Programs the state of AWCACHE[3,1:0]a when the DMAC writes the destination data.</p> <p>Bit [27] 0 = AWCACHE[3] is LOW 1 = AWCACHE[3] is HIGH.</p> <p>Bit [26] 0 = AWCACHE[1] is LOW 1 = AWCACHE[1] is HIGH.</p> <p>Bit [25] 0 = AWCACHE[0] is LOW 1 = AWCACHE[0] is HIGH.</p> <p>Note</p> <p>AWCACHE[2] is tied LOW by the DMAC.</p> <p>Setting AWCACHE[3,1]=b10 violates the AXI protocol. See the AMBA AXI Protocol Specification.</p>
dst_prot_ctrl	24:22	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of AWPROT[2:0]a when the DMAC writes the destination data.</p> <p>Bit [24] 0 = AWPROT[2] is LOW 1 = AWPROT[2] is HIGH.</p> <p>Bit [23] 0 = AWPROT[1] is LOW 1 = AWPROT[1] is HIGH.</p> <p>Bit [22] 0 = AWPROT[0] is LOW 1 = AWPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program AWPROT[1] LOW, that is, a secure access. If</p> <p>a DMA channel in the Non-secure state attempts to set AWPROT[1] LOW, then the DMA channel aborts</p>

Field Name	Bits	Type	Reset Value	Description
dst_burst_len	21:18	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it writes the destination data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers . . . b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction is the product of dst_burst_len and dst_burst_size.</p> <p>Note These bits control the state of AWLEN[3:0].</p>
dst_burst_size	17:15	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC writes to the destination:</p> <p>b000 = writes 1 byte per beat b001 = writes 2 bytes per beat b010 = writes 4 bytes per beat b011 = writes 8 bytes per beat b100 = writes 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction is the product of dst_burst_len and dst_burst_size.</p> <p>Note These bits control the state of AWSIZE[2:0].</p>
dst_inc	14	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it writes the destination data:</p> <p>0 = Fixed-address burst. The DMAC signals AWBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals AWBURST[0] HIGH.</p>

Field Name	Bits	Type	Reset Value	Description
src_cache_ctrl	13:11	sro,ns sraz,n snsro	0x0	<p>Set the bits to control the state of ARCACHE[2:0]a when the DMAC reads the source data.</p> <p>Bit [13] 0 = ARCACHE[2] is LOW 1 = ARCACHE[2] is HIGH.</p> <p>Bit [12] 0 = ARCACHE[1] is LOW 1 = ARCACHE[1] is HIGH.</p> <p>Bit [11] 0 = ARCACHE[0] is LOW 1 = ARCACHE[0] is HIGH.</p> <p>Note</p> <p>The DMAC ties ARCACHE[3] LOW.</p> <p>Setting ARCACHE[2:1]=b10 violates the AXI protocol.</p>
src_prot_ctrl	10:8	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of ARPROT[2:0]a when the DMAC reads the source data.</p> <p>Bit [10] 0 = ARPROT[2] is LOW 1 = ARPROT[2] is HIGH.</p> <p>Bit [9] 0 = ARPROT[1] is LOW 1 = ARPROT[1] is HIGH.</p> <p>Bit [8] 0 = ARPROT[0] is LOW 1 = ARPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program ARPROT[1] LOW, that is, a secure access. If a</p> <p>DMA channel in the Non-secure state attempts to set ARPROT[1] LOW, the DMA channel aborts.</p>
src_burst_len	7:4	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it reads</p> <p>the source data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers . . . b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction</p> <p>is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
src_burst_size	3:1	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC reads from the source:</p> <p>b000 = reads 1 byte per beat b001 = reads 2 bytes per beat b010 = reads 4 bytes per beat b011 = reads 8 bytes per beat b100 = reads 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARSIZE[2:0].</p>
src_inc	0	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it reads the source data:</p> <p>0 = Fixed-address burst. The DMAC signals ARBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals ARBURST[0] HIGH.</p>

Register ([dmac](#)) LC0_0

Name	LC0_0
Relative Address	0x0000040C
Absolute Address	dmac0_ns: 0xF800440C dmac0_s: 0xF800340C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 0 for DMA channel 0

Register LC0_0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter zero for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter zero

Register ([dmac](#)) LC1_0

Name	LC1_0
Relative Address	0x00000410
Absolute Address	dmac0_ns: 0xF8004410 dmac0_s: 0xF8003410
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 1 for DMA channel 0

Register LC1_0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter one for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter one.

Register ([dmac](#)) SAR1

Name	SAR1
Software Name	XDmaPs_SA_n_OFFSET(1)
Relative Address	0x00000420
Absolute Address	dmac0_ns: 0xF8004420 dmac0_s: 0xF8003420
Width	32 bits

Access Type	mixed
Reset Value	0x00000000
Description	source address for DMA channel 1

Register SAR1 Details

Field Name	Bits	Type	Reset Value	Description
src_addr	31:0	sro,ns sraz,n snsro	0x0	Address of the source data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) DAR1

Name	DAR1
Software Name	XDmaPs_DA_n_OFFSET(1)
Relative Address	0x00000424
Absolute Address	dmac0_ns: 0xF8004424 dmac0_s: 0xF8003424
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	destination address for DMA channel 1

Register DAR1 Details

Field Name	Bits	Type	Reset Value	Description
dest_addr	31:0	sro,ns sraz,n snsro	0x0	Address for the destination data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) CCR1

Name	CCR1
Software Name	XDmaPs_CC_n_OFFSET(1)
Relative Address	0x00000428
Absolute Address	dmac0_ns: 0xF8004428 dmac0_s: 0xF8003428
Width	32 bits
Access Type	mixed

Reset Value dmac0_ns: 0x00000000
 dmac0_s: 0x00800200

Description channel control for DMA channel 1

Register CCR1 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	rud	0x0	reserved, read undefined
endian_swap_size	30:28	sro,ns sraz,n snsro	0x0	<p>Table 3-22 on page 3-29 defines whether data can be swapped between little-endian (LE) and byte-invariant big-endian (BE-8) formats, and if so, also defines the natural width of the data independently of the source and destination transaction sizes. This enables unaligned data streams to use the full bus-width, and to be correctly transformed, irrespective of the source and destination address alignments. The format is identical to AxsIZE, except that b000 indicates that no swap must occur.</p> <p>Endian swap size Description</p> <p>b000 No swap, 8-bit data</p> <p>b001 Swap bytes within 16-bit data</p> <p>b010 Swap bytes within 32-bit data</p> <p>b011 Swap bytes within 64-bit data</p> <p>b100 Swap bytes within 128-bit data</p> <p>b101 Reserved</p> <p>b110 Reserved</p> <p>b111 Reserved</p>
dst_cache_ctrl	27:25	sro,ns sraz,n snsro	0x0	<p>Programs the state of AWCACHE[3,1:0]a when the DMAC writes the destination data.</p> <p>Bit [27] 0 = AWCACHE[3] is LOW 1 = AWCACHE[3] is HIGH.</p> <p>Bit [26] 0 = AWCACHE[1] is LOW 1 = AWCACHE[1] is HIGH.</p> <p>Bit [25] 0 = AWCACHE[0] is LOW 1 = AWCACHE[0] is HIGH.</p> <p>Note</p> <p>AWCACHE[2] is tied LOW by the DMAC.</p> <p>Setting AWCACHE[3,1]=b10 violates the AXI protocol. See the AMBA AXI Protocol Specification.</p>

Field Name	Bits	Type	Reset Value	Description
dst_prot_ctrl	24:22	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of AWPROT[2:0]a when the DMAC writes the destination data.</p> <p>Bit [24] 0 = AWPROT[2] is LOW 1 = AWPROT[2] is HIGH.</p> <p>Bit [23] 0 = AWPROT[1] is LOW 1 = AWPROT[1] is HIGH.</p> <p>Bit [22] 0 = AWPROT[0] is LOW 1 = AWPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program AWPROT[1] LOW, that is, a secure access. If</p> <p>a DMA channel in the Non-secure state attempts to set AWPROT[1] LOW, then the DMA channel aborts</p>
dst_burst_len	21:18	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it writes</p> <p>the destination data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers</p> <p>.</p> <p>.</p> <p>.</p> <p>b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction</p> <p>is the product of dst_burst_len and dst_burst_size.</p> <p>Note</p> <p>These bits control the state of AWLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
dst_burst_size	17:15	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC writes to the destination:</p> <p>b000 = writes 1 byte per beat b001 = writes 2 bytes per beat b010 = writes 4 bytes per beat b011 = writes 8 bytes per beat b100 = writes 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction</p> <p>is the product of dst_burst_len and dst_burst_size.</p> <p>Note</p> <p>These bits control the state of AWSIZE[2:0].</p>
dst_inc	14	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it writes the destination data:</p> <p>0 = Fixed-address burst. The DMAC signals AWBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals AWBURST[0] HIGH.</p>
src_cache_ctrl	13:11	sro,ns sraz,n snsro	0x0	<p>Set the bits to control the state of ARCACHE[2:0]a when the DMAC reads the source data.</p> <p>Bit [13] 0 = ARCACHE[2] is LOW 1 = ARCACHE[2] is HIGH.</p> <p>Bit [12] 0 = ARCACHE[1] is LOW 1 = ARCACHE[1] is HIGH.</p> <p>Bit [11] 0 = ARCACHE[0] is LOW 1 = ARCACHE[0] is HIGH.</p> <p>Note</p> <p>The DMAC ties ARCACHE[3] LOW.</p> <p>Setting ARCACHE[2:1]=b10 violates the AXI protocol.</p>

Field Name	Bits	Type	Reset Value	Description
src_prot_ctrl	10:8	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of ARPROT[2:0]a when the DMAC reads the source data.</p> <p>Bit [10] 0 = ARPROT[2] is LOW 1 = ARPROT[2] is HIGH.</p> <p>Bit [9] 0 = ARPROT[1] is LOW 1 = ARPROT[1] is HIGH.</p> <p>Bit [8] 0 = ARPROT[0] is LOW 1 = ARPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program ARPROT[1] LOW, that is, a secure access. If a</p> <p>DMA channel in the Non-secure state attempts to set ARPROT[1] LOW, the DMA channel aborts.</p>
src_burst_len	7:4	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it reads the source data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers . . . b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
src_burst_size	3:1	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC reads from the source:</p> <p>b000 = reads 1 byte per beat b001 = reads 2 bytes per beat b010 = reads 4 bytes per beat b011 = reads 8 bytes per beat b100 = reads 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARSIZE[2:0].</p>
src_inc	0	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it reads the source data:</p> <p>0 = Fixed-address burst. The DMAC signals ARBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals ARBURST[0] HIGH.</p>

Register ([dmac](#)) LC0_1

Name	LC0_1
Software Name	XDmaPs_LC0_n_OFFSET(1)
Relative Address	0x0000042C
Absolute Address	dmac0_ns: 0xF800442C dmac0_s: 0xF800342C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 0 for DMA channel 1

Register LC0_1 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter zero for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter zero

Register ([dmac](#)) LC1_1

Name	LC1_1
Software Name	XDmaPs_LC1_n_OFFSET(1)
Relative Address	0x00000430
Absolute Address	dmac0_ns: 0xF8004430 dmac0_s: 0xF8003430
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 1 for DMA channel 1

Register LC1_1 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter one for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter one.

Register ([dmac](#)) SAR2

Name	SAR2
Software Name	XDmaPs_SA_n_OFFSET(2)
Relative Address	0x00000440
Absolute Address	dmac0_ns: 0xF8004440 dmac0_s: 0xF8003440

Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	source address for DMA channel 2

Register SAR2 Details

Field Name	Bits	Type	Reset Value	Description
src_addr	31:0	sro,ns sraz,n snsro	0x0	Address of the source data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) DAR2

Name	DAR2
Software Name	XDmaPs_DA_n_OFFSET(2)
Relative Address	0x00000444
Absolute Address	dmac0_ns: 0xF8004444 dmac0_s: 0xF8003444
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	destination address for DMA channel 2

Register DAR2 Details

Field Name	Bits	Type	Reset Value	Description
dest_addr	31:0	sro,ns sraz,n snsro	0x0	Address for the destination data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) CCR2

Name	CCR2
Software Name	XDmaPs_CC_n_OFFSET(2)
Relative Address	0x00000448
Absolute Address	dmac0_ns: 0xF8004448 dmac0_s: 0xF8003448
Width	32 bits
Access Type	mixed

Reset Value dmac0_ns: 0x00000000
 dmac0_s: 0x00800200

Description channel control for DMA channel 2

Register CCR2 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	rud	0x0	reserved, read undefined
endian_swap_size	30:28	sro,ns sraz,n snsro	0x0	<p>Table 3-22 on page 3-29 defines whether data can be swapped between little-endian (LE) and byte-invariant big-endian (BE-8) formats, and if so, also defines the natural width of the data independently of the source and destination transaction sizes. This enables unaligned data streams to use the full bus-width, and to be correctly transformed, irrespective of the source and destination address alignments. The format is identical to AxsIZE, except that b000 indicates that no swap must occur.</p> <p>Endian swap size Description</p> <p>b000 No swap, 8-bit data</p> <p>b001 Swap bytes within 16-bit data</p> <p>b010 Swap bytes within 32-bit data</p> <p>b011 Swap bytes within 64-bit data</p> <p>b100 Swap bytes within 128-bit data</p> <p>b101 Reserved</p> <p>b110 Reserved</p> <p>b111 Reserved</p>
dst_cache_ctrl	27:25	sro,ns sraz,n snsro	0x0	<p>Programs the state of AWCACHE[3,1:0]a when the DMAC writes the destination data.</p> <p>Bit [27] 0 = AWCACHE[3] is LOW 1 = AWCACHE[3] is HIGH.</p> <p>Bit [26] 0 = AWCACHE[1] is LOW 1 = AWCACHE[1] is HIGH.</p> <p>Bit [25] 0 = AWCACHE[0] is LOW 1 = AWCACHE[0] is HIGH.</p> <p>Note</p> <p>AWCACHE[2] is tied LOW by the DMAC.</p> <p>Setting AWCACHE[3,1]=b10 violates the AXI protocol. See the AMBA AXI Protocol Specification.</p>

Field Name	Bits	Type	Reset Value	Description
dst_prot_ctrl	24:22	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of AWPROT[2:0]a when the DMAC writes the destination data.</p> <p>Bit [24] 0 = AWPROT[2] is LOW 1 = AWPROT[2] is HIGH.</p> <p>Bit [23] 0 = AWPROT[1] is LOW 1 = AWPROT[1] is HIGH.</p> <p>Bit [22] 0 = AWPROT[0] is LOW 1 = AWPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program AWPROT[1] LOW, that is, a secure access. If</p> <p>a DMA channel in the Non-secure state attempts to set AWPROT[1] LOW, then the DMA channel aborts</p>
dst_burst_len	21:18	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it writes</p> <p>the destination data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers</p> <p>· · ·</p> <p>b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction</p> <p>is the product of dst_burst_len and dst_burst_size.</p> <p>Note</p> <p>These bits control the state of AWLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
dst_burst_size	17:15	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC writes to the destination:</p> <p>b000 = writes 1 byte per beat b001 = writes 2 bytes per beat b010 = writes 4 bytes per beat b011 = writes 8 bytes per beat b100 = writes 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction is the product of dst_burst_len and dst_burst_size.</p> <p>Note These bits control the state of AWSIZE[2:0].</p>
dst_inc	14	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it writes the destination data:</p> <p>0 = Fixed-address burst. The DMAC signals AWBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals AWBURST[0] HIGH.</p>
src_cache_ctrl	13:11	sro,ns sraz,n snsro	0x0	<p>Set the bits to control the state of ARCACHE[2:0]a when the DMAC reads the source data.</p> <p>Bit [13] 0 = ARCACHE[2] is LOW 1 = ARCACHE[2] is HIGH.</p> <p>Bit [12] 0 = ARCACHE[1] is LOW 1 = ARCACHE[1] is HIGH.</p> <p>Bit [11] 0 = ARCACHE[0] is LOW 1 = ARCACHE[0] is HIGH.</p> <p>Note The DMAC ties ARCACHE[3] LOW. Setting ARCACHE[2:1]=b10 violates the AXI protocol.</p>

Field Name	Bits	Type	Reset Value	Description
src_prot_ctrl	10:8	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of ARPROT[2:0]a when the DMAC reads the source data.</p> <p>Bit [10] 0 = ARPROT[2] is LOW 1 = ARPROT[2] is HIGH.</p> <p>Bit [9] 0 = ARPROT[1] is LOW 1 = ARPROT[1] is HIGH.</p> <p>Bit [8] 0 = ARPROT[0] is LOW 1 = ARPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program ARPROT[1] LOW, that is, a secure access. If a</p> <p>DMA channel in the Non-secure state attempts to set ARPROT[1] LOW, the DMA channel aborts.</p>
src_burst_len	7:4	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it reads</p> <p>the source data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers . . . b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction</p> <p>is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
src_burst_size	3:1	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC reads from the source:</p> <p>b000 = reads 1 byte per beat b001 = reads 2 bytes per beat b010 = reads 4 bytes per beat b011 = reads 8 bytes per beat b100 = reads 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARSIZE[2:0].</p>
src_inc	0	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it reads the source data:</p> <p>0 = Fixed-address burst. The DMAC signals ARBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals ARBURST[0] HIGH.</p>

Register ([dmac](#)) LC0_2

Name	LC0_2
Software Name	XDmaPs_LC0_n_OFFSET(2)
Relative Address	0x0000044C
Absolute Address	dmac0_ns: 0xF800444C dmac0_s: 0xF800344C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 0 for DMA channel 2

Register LC0_2 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter zero for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter zero

Register ([dmac](#)) LC1_2

Name	LC1_2
Software Name	XDmaPs_LC1_n_OFFSET(2)
Relative Address	0x00000450
Absolute Address	dmac0_ns: 0xF8004450 dmac0_s: 0xF8003450
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 1 for DMA channel 2

Register LC1_2 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter one for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter one.

Register ([dmac](#)) SAR3

Name	SAR3
Software Name	XDmaPs_SA_n_OFFSET(3)
Relative Address	0x00000460
Absolute Address	dmac0_ns: 0xF8004460 dmac0_s: 0xF8003460

Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	source address for DMA channel 3

Register SAR3 Details

Field Name	Bits	Type	Reset Value	Description
src_addr	31:0	sro,ns sraz,n snsro	0x0	Address of the source data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) DAR3

Name	DAR3
Software Name	XDmaPs_DA_n_OFFSET(3)
Relative Address	0x00000464
Absolute Address	dmac0_ns: 0xF8004464 dmac0_s: 0xF8003464
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	destination address for DMA channel 3

Register DAR3 Details

Field Name	Bits	Type	Reset Value	Description
dest_addr	31:0	sro,ns sraz,n snsro	0x0	Address for the destination data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) CCR3

Name	CCR3
Software Name	XDmaPs_CC_n_OFFSET(3)
Relative Address	0x00000468
Absolute Address	dmac0_ns: 0xF8004468 dmac0_s: 0xF8003468
Width	32 bits
Access Type	mixed

Reset Value dmac0_ns: 0x00000000
 dmac0_s: 0x00800200

Description channel control for DMA channel 3

Register CCR3 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	rud	0x0	reserved, read undefined
endian_swap_size	30:28	sro,ns sraz,n snsro	0x0	<p>Table 3-22 on page 3-29 defines whether data can be swapped between little-endian (LE) and byte-invariant big-endian (BE-8) formats, and if so, also defines the natural width of the data independently of the source and destination transaction sizes. This enables unaligned data streams to use the full bus-width, and to be correctly transformed, irrespective of the source and destination address alignments. The format is identical to AxsIZE, except that b000 indicates that no swap must occur.</p> <p>Endian swap size Description</p> <p>b000 No swap, 8-bit data</p> <p>b001 Swap bytes within 16-bit data</p> <p>b010 Swap bytes within 32-bit data</p> <p>b011 Swap bytes within 64-bit data</p> <p>b100 Swap bytes within 128-bit data</p> <p>b101 Reserved</p> <p>b110 Reserved</p> <p>b111 Reserved</p>
dst_cache_ctrl	27:25	sro,ns sraz,n snsro	0x0	<p>Programs the state of AWCACHE[3,1:0]a when the DMAC writes the destination data.</p> <p>Bit [27] 0 = AWCACHE[3] is LOW 1 = AWCACHE[3] is HIGH.</p> <p>Bit [26] 0 = AWCACHE[1] is LOW 1 = AWCACHE[1] is HIGH.</p> <p>Bit [25] 0 = AWCACHE[0] is LOW 1 = AWCACHE[0] is HIGH.</p> <p>Note</p> <p>AWCACHE[2] is tied LOW by the DMAC.</p> <p>Setting AWCACHE[3,1]=b10 violates the AXI protocol. See the AMBA AXI Protocol Specification.</p>

Field Name	Bits	Type	Reset Value	Description
dst_prot_ctrl	24:22	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of AWPROT[2:0]a when the DMAC writes the destination data.</p> <p>Bit [24] 0 = AWPROT[2] is LOW 1 = AWPROT[2] is HIGH.</p> <p>Bit [23] 0 = AWPROT[1] is LOW 1 = AWPROT[1] is HIGH.</p> <p>Bit [22] 0 = AWPROT[0] is LOW 1 = AWPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program AWPROT[1] LOW, that is, a secure access. If</p> <p>a DMA channel in the Non-secure state attempts to set AWPROT[1] LOW, then the DMA channel aborts</p>
dst_burst_len	21:18	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it writes</p> <p>the destination data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers</p> <p>· · ·</p> <p>b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction</p> <p>is the product of dst_burst_len and dst_burst_size.</p> <p>Note</p> <p>These bits control the state of AWLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
dst_burst_size	17:15	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC writes to the destination:</p> <p>b000 = writes 1 byte per beat b001 = writes 2 bytes per beat b010 = writes 4 bytes per beat b011 = writes 8 bytes per beat b100 = writes 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction is the product of dst_burst_len and dst_burst_size.</p> <p>Note These bits control the state of AWSIZE[2:0].</p>
dst_inc	14	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it writes the destination data:</p> <p>0 = Fixed-address burst. The DMAC signals AWBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals AWBURST[0] HIGH.</p>
src_cache_ctrl	13:11	sro,ns sraz,n snsro	0x0	<p>Set the bits to control the state of ARCACHE[2:0]a when the DMAC reads the source data.</p> <p>Bit [13] 0 = ARCACHE[2] is LOW 1 = ARCACHE[2] is HIGH.</p> <p>Bit [12] 0 = ARCACHE[1] is LOW 1 = ARCACHE[1] is HIGH.</p> <p>Bit [11] 0 = ARCACHE[0] is LOW 1 = ARCACHE[0] is HIGH.</p> <p>Note The DMAC ties ARCACHE[3] LOW. Setting ARCACHE[2:1]=b10 violates the AXI protocol.</p>

Field Name	Bits	Type	Reset Value	Description
src_prot_ctrl	10:8	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of ARPROT[2:0]a when the DMAC reads the source data.</p> <p>Bit [10] 0 = ARPROT[2] is LOW 1 = ARPROT[2] is HIGH.</p> <p>Bit [9] 0 = ARPROT[1] is LOW 1 = ARPROT[1] is HIGH.</p> <p>Bit [8] 0 = ARPROT[0] is LOW 1 = ARPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program ARPROT[1] LOW, that is, a secure access. If a</p> <p>DMA channel in the Non-secure state attempts to set ARPROT[1] LOW, the DMA channel aborts.</p>
src_burst_len	7:4	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it reads</p> <p>the source data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers . . . b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction</p> <p>is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
src_burst_size	3:1	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC reads from the source:</p> <p>b000 = reads 1 byte per beat b001 = reads 2 bytes per beat b010 = reads 4 bytes per beat b011 = reads 8 bytes per beat b100 = reads 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARSIZE[2:0].</p>
src_inc	0	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it reads the source data:</p> <p>0 = Fixed-address burst. The DMAC signals ARBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals ARBURST[0] HIGH.</p>

Register ([dmac](#)) LC0_3

Name	LC0_3
Software Name	XDmaPs_LC0_n_OFFSET(3)
Relative Address	0x0000046C
Absolute Address	dmac0_ns: 0xF800446C dmac0_s: 0xF800346C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 0 for DMA channel 3

Register LC0_3 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter zero for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter zero

Register ([dmac](#)) LC1_3

Name	LC1_3
Software Name	XDmaPs_LC1_n_OFFSET(3)
Relative Address	0x00000470
Absolute Address	dmac0_ns: 0xF8004470 dmac0_s: 0xF8003470
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 1 for DMA channel 3

Register LC1_3 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter one for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter one.

Register ([dmac](#)) SAR4

Name	SAR4
Software Name	XDmaPs_SA_n_OFFSET(4)
Relative Address	0x00000480
Absolute Address	dmac0_ns: 0xF8004480 dmac0_s: 0xF8003480

Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	source address for DMA channel 4

Register SAR4 Details

Field Name	Bits	Type	Reset Value	Description
src_addr	31:0	sro,ns sraz,n snsro	0x0	Address of the source data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) DAR4

Name	DAR4
Software Name	XDmaPs_DA_n_OFFSET(4)
Relative Address	0x00000484
Absolute Address	dmac0_ns: 0xF8004484 dmac0_s: 0xF8003484
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	destination address for DMA channel 4

Register DAR4 Details

Field Name	Bits	Type	Reset Value	Description
dest_addr	31:0	sro,ns sraz,n snsro	0x0	Address for the destination data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) CCR4

Name	CCR4
Software Name	XDmaPs_CC_n_OFFSET(4)
Relative Address	0x00000488
Absolute Address	dmac0_ns: 0xF8004488 dmac0_s: 0xF8003488
Width	32 bits
Access Type	mixed

Reset Value dmac0_ns: 0x00000000
 dmac0_s: 0x00800200

Description channel control for DMA channel 4

Register CCR4 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	rud	0x0	reserved, read undefined
endian_swap_size	30:28	sro,ns sraz,n snsro	0x0	<p>Table 3-22 on page 3-29 defines whether data can be swapped between little-endian (LE) and byte-invariant big-endian (BE-8) formats, and if so, also defines the natural width of the data independently of the source and destination transaction sizes. This enables unaligned data streams to use the full bus-width, and to be correctly transformed, irrespective of the source and destination address alignments. The format is identical to AxsIZE, except that b000 indicates that no swap must occur.</p> <p>Endian swap size Description</p> <p>b000 No swap, 8-bit data</p> <p>b001 Swap bytes within 16-bit data</p> <p>b010 Swap bytes within 32-bit data</p> <p>b011 Swap bytes within 64-bit data</p> <p>b100 Swap bytes within 128-bit data</p> <p>b101 Reserved</p> <p>b110 Reserved</p> <p>b111 Reserved</p>
dst_cache_ctrl	27:25	sro,ns sraz,n snsro	0x0	<p>Programs the state of AWCACHE[3,1:0]a when the DMAC writes the destination data.</p> <p>Bit [27] 0 = AWCACHE[3] is LOW 1 = AWCACHE[3] is HIGH.</p> <p>Bit [26] 0 = AWCACHE[1] is LOW 1 = AWCACHE[1] is HIGH.</p> <p>Bit [25] 0 = AWCACHE[0] is LOW 1 = AWCACHE[0] is HIGH.</p> <p>Note</p> <p>AWCACHE[2] is tied LOW by the DMAC.</p> <p>Setting AWCACHE[3,1]=b10 violates the AXI protocol. See the AMBA AXI Protocol Specification.</p>

Field Name	Bits	Type	Reset Value	Description
dst_prot_ctrl	24:22	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of AWPROT[2:0]a when the DMAC writes the destination data.</p> <p>Bit [24] 0 = AWPROT[2] is LOW 1 = AWPROT[2] is HIGH.</p> <p>Bit [23] 0 = AWPROT[1] is LOW 1 = AWPROT[1] is HIGH.</p> <p>Bit [22] 0 = AWPROT[0] is LOW 1 = AWPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program AWPROT[1] LOW, that is, a secure access. If</p> <p>a DMA channel in the Non-secure state attempts to set AWPROT[1] LOW, then the DMA channel aborts</p>
dst_burst_len	21:18	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it writes</p> <p>the destination data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers</p> <p>.</p> <p>.</p> <p>.</p> <p>b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction</p> <p>is the product of dst_burst_len and dst_burst_size.</p> <p>Note</p> <p>These bits control the state of AWLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
dst_burst_size	17:15	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC writes to the destination:</p> <p>b000 = writes 1 byte per beat b001 = writes 2 bytes per beat b010 = writes 4 bytes per beat b011 = writes 8 bytes per beat b100 = writes 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction is the product of dst_burst_len and dst_burst_size.</p> <p>Note These bits control the state of AWSIZE[2:0].</p>
dst_inc	14	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it writes the destination data:</p> <p>0 = Fixed-address burst. The DMAC signals AWBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals AWBURST[0] HIGH.</p>
src_cache_ctrl	13:11	sro,ns sraz,n snsro	0x0	<p>Set the bits to control the state of ARCACHE[2:0]a when the DMAC reads the source data.</p> <p>Bit [13] 0 = ARCACHE[2] is LOW 1 = ARCACHE[2] is HIGH.</p> <p>Bit [12] 0 = ARCACHE[1] is LOW 1 = ARCACHE[1] is HIGH.</p> <p>Bit [11] 0 = ARCACHE[0] is LOW 1 = ARCACHE[0] is HIGH.</p> <p>Note The DMAC ties ARCACHE[3] LOW. Setting ARCACHE[2:1]=b10 violates the AXI protocol.</p>

Field Name	Bits	Type	Reset Value	Description
src_prot_ctrl	10:8	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of ARPROT[2:0]a when the DMAC reads the source data.</p> <p>Bit [10] 0 = ARPROT[2] is LOW 1 = ARPROT[2] is HIGH.</p> <p>Bit [9] 0 = ARPROT[1] is LOW 1 = ARPROT[1] is HIGH.</p> <p>Bit [8] 0 = ARPROT[0] is LOW 1 = ARPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program ARPROT[1] LOW, that is, a secure access. If a</p> <p>DMA channel in the Non-secure state attempts to set ARPROT[1] LOW, the DMA channel aborts.</p>
src_burst_len	7:4	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it reads</p> <p>the source data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers . . . b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction</p> <p>is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
src_burst_size	3:1	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC reads from the source:</p> <p>b000 = reads 1 byte per beat b001 = reads 2 bytes per beat b010 = reads 4 bytes per beat b011 = reads 8 bytes per beat b100 = reads 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size.</p> <p>Note These bits control the state of ARSIZE[2:0].</p>
src_inc	0	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it reads the source data:</p> <p>0 = Fixed-address burst. The DMAC signals ARBURST[0] LOW. 1 = Incrementing-address burst. The DMAC signals ARBURST[0] HIGH.</p>

Register ([dmac](#)) LC0_4

Name	LC0_4
Software Name	XDmaPs_LC0_n_OFFSET(4)
Relative Address	0x0000048C
Absolute Address	dmac0_ns: 0xF800448C dmac0_s: 0xF800348C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 0 for DMA channel 4

Register LC0_4 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter zero for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter zero

Register ([dmac](#)) LC1_4

Name	LC1_4
Software Name	XDmaPs_LC1_n_OFFSET(4)
Relative Address	0x00000490
Absolute Address	dmac0_ns: 0xF8004490 dmac0_s: 0xF8003490
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 1 for DMA channel 4

Register LC1_4 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter one for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter one.

Register ([dmac](#)) SAR5

Name	SAR5
Software Name	XDmaPs_SA_n_OFFSET(5)
Relative Address	0x000004A0
Absolute Address	dmac0_ns: 0xF80044A0 dmac0_s: 0xF80034A0

Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	source address for DMA channel 5

Register SAR5 Details

Field Name	Bits	Type	Reset Value	Description
src_addr	31:0	sro,ns sraz,n snsro	0x0	Address of the source data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) DAR5

Name	DAR5
Software Name	XDmaPs_DA_n_OFFSET(5)
Relative Address	0x000004A4
Absolute Address	dmac0_ns: 0xF80044A4 dmac0_s: 0xF80034A4
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	destination address for DMA channel 5

Register DAR5 Details

Field Name	Bits	Type	Reset Value	Description
dest_addr	31:0	sro,ns sraz,n snsro	0x0	Address for the destination data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) CCR5

Name	CCR5
Software Name	XDmaPs_CC_n_OFFSET(5)
Relative Address	0x000004A8
Absolute Address	dmac0_ns: 0xF80044A8 dmac0_s: 0xF80034A8
Width	32 bits
Access Type	mixed

Reset Value dmac0_ns: 0x00000000
 dmac0_s: 0x00800200

Description channel control for DMA channel 5

Register CCR5 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	rud	0x0	reserved, read undefined
endian_swap_size	30:28	sro,ns sraz,n snsro	0x0	<p>Table 3-22 on page 3-29 defines whether data can be swapped between little-endian (LE) and byte-invariant big-endian (BE-8) formats, and if so, also defines the natural width of the data independently of the source and destination transaction sizes. This enables unaligned data streams to use the full bus-width, and to be correctly transformed, irrespective of the source and destination address alignments. The format is identical to AxsIZE, except that b000 indicates that no swap must occur.</p> <p>Endian swap size Description</p> <p>b000 No swap, 8-bit data</p> <p>b001 Swap bytes within 16-bit data</p> <p>b010 Swap bytes within 32-bit data</p> <p>b011 Swap bytes within 64-bit data</p> <p>b100 Swap bytes within 128-bit data</p> <p>b101 Reserved</p> <p>b110 Reserved</p> <p>b111 Reserved</p>
dst_cache_ctrl	27:25	sro,ns sraz,n snsro	0x0	<p>Programs the state of AWCACHE[3,1:0]a when the DMAC writes the destination data.</p> <p>Bit [27] 0 = AWCACHE[3] is LOW 1 = AWCACHE[3] is HIGH.</p> <p>Bit [26] 0 = AWCACHE[1] is LOW 1 = AWCACHE[1] is HIGH.</p> <p>Bit [25] 0 = AWCACHE[0] is LOW 1 = AWCACHE[0] is HIGH.</p> <p>Note</p> <p>AWCACHE[2] is tied LOW by the DMAC.</p> <p>Setting AWCACHE[3,1]=b10 violates the AXI protocol. See the AMBA AXI Protocol Specification.</p>

Field Name	Bits	Type	Reset Value	Description
dst_prot_ctrl	24:22	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of AWPROT[2:0]a when the DMAC writes the destination data.</p> <p>Bit [24] 0 = AWPROT[2] is LOW 1 = AWPROT[2] is HIGH.</p> <p>Bit [23] 0 = AWPROT[1] is LOW 1 = AWPROT[1] is HIGH.</p> <p>Bit [22] 0 = AWPROT[0] is LOW 1 = AWPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program AWPROT[1] LOW, that is, a secure access. If</p> <p>a DMA channel in the Non-secure state attempts to set AWPROT[1] LOW, then the DMA channel aborts</p>
dst_burst_len	21:18	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it writes</p> <p>the destination data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers</p> <p>· · ·</p> <p>b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction</p> <p>is the product of dst_burst_len and dst_burst_size.</p> <p>Note</p> <p>These bits control the state of AWLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
dst_burst_size	17:15	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC writes to the destination:</p> <p>b000 = writes 1 byte per beat b001 = writes 2 bytes per beat b010 = writes 4 bytes per beat b011 = writes 8 bytes per beat b100 = writes 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction</p> <p>is the product of dst_burst_len and dst_burst_size.</p> <p>Note</p> <p>These bits control the state of AWSIZE[2:0].</p>
dst_inc	14	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it writes the destination data:</p> <p>0 = Fixed-address burst. The DMAC signals AWBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals AWBURST[0] HIGH.</p>
src_cache_ctrl	13:11	sro,ns sraz,n snsro	0x0	<p>Set the bits to control the state of ARCACHE[2:0]a when the DMAC reads the source data.</p> <p>Bit [13] 0 = ARCACHE[2] is LOW 1 = ARCACHE[2] is HIGH.</p> <p>Bit [12] 0 = ARCACHE[1] is LOW 1 = ARCACHE[1] is HIGH.</p> <p>Bit [11] 0 = ARCACHE[0] is LOW 1 = ARCACHE[0] is HIGH.</p> <p>Note</p> <p>The DMAC ties ARCACHE[3] LOW.</p> <p>Setting ARCACHE[2:1]=b10 violates the AXI protocol.</p>

Field Name	Bits	Type	Reset Value	Description
src_prot_ctrl	10:8	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of ARPROT[2:0]a when the DMAC reads the source data.</p> <p>Bit [10] 0 = ARPROT[2] is LOW 1 = ARPROT[2] is HIGH.</p> <p>Bit [9] 0 = ARPROT[1] is LOW 1 = ARPROT[1] is HIGH.</p> <p>Bit [8] 0 = ARPROT[0] is LOW 1 = ARPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program ARPROT[1] LOW, that is, a secure access. If a</p> <p>DMA channel in the Non-secure state attempts to set ARPROT[1] LOW, the DMA channel aborts.</p>
src_burst_len	7:4	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it reads the source data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers . . . b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
src_burst_size	3:1	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC reads from the source:</p> <p>b000 = reads 1 byte per beat b001 = reads 2 bytes per beat b010 = reads 4 bytes per beat b011 = reads 8 bytes per beat b100 = reads 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARSIZE[2:0].</p>
src_inc	0	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it reads the source data:</p> <p>0 = Fixed-address burst. The DMAC signals ARBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals ARBURST[0] HIGH.</p>

Register ([dmac](#)) LC0_5

Name	LC0_5
Software Name	XDmaPs_LC0_n_OFFSET(5)
Relative Address	0x000004AC
Absolute Address	dmac0_ns: 0xF80044AC dmac0_s: 0xF80034AC
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 0 for DMA channel 5

Register LC0_5 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter zero for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter zero

Register ([dmac](#)) LC1_5

Name	LC1_5
Software Name	XDmaPs_LC1_n_OFFSET(5)
Relative Address	0x000004B0
Absolute Address	dmac0_ns: 0xF80044B0 dmac0_s: 0xF80034B0
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 1 for DMA channel 5

Register LC1_5 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter one for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter one.

Register ([dmac](#)) SAR6

Name	SAR6
Software Name	XDmaPs_SA_n_OFFSET(6)
Relative Address	0x000004C0
Absolute Address	dmac0_ns: 0xF80044C0 dmac0_s: 0xF80034C0

Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	source address for DMA channel 6

Register SAR6 Details

Field Name	Bits	Type	Reset Value	Description
src_addr	31:0	sro,ns sraz,n snsro	0x0	Address of the source data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) DAR6

Name	DAR6
Software Name	XDmaPs_DA_n_OFFSET(6)
Relative Address	0x000004C4
Absolute Address	dmac0_ns: 0xF80044C4 dmac0_s: 0xF80034C4
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	destination address for DMA channel 6

Register DAR6 Details

Field Name	Bits	Type	Reset Value	Description
dest_addr	31:0	sro,ns sraz,n snsro	0x0	Address for the destination data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) CCR6

Name	CCR6
Software Name	XDmaPs_CC_n_OFFSET(6)
Relative Address	0x000004C8
Absolute Address	dmac0_ns: 0xF80044C8 dmac0_s: 0xF80034C8
Width	32 bits
Access Type	mixed

Reset Value dmac0_ns: 0x00000000
 dmac0_s: 0x00800200

Description channel control for DMA channel 6

Register CCR6 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	rud	0x0	reserved, read undefined
endian_swap_size	30:28	sro,ns sraz,n snsro	0x0	<p>Table 3-22 on page 3-29 defines whether data can be swapped between little-endian (LE) and byte-invariant big-endian (BE-8) formats, and if so, also defines the natural width of the data independently of the source and destination transaction sizes. This enables unaligned data streams to use the full bus-width, and to be correctly transformed, irrespective of the source and destination address alignments. The format is identical to AxsIZE, except that b000 indicates that no swap must occur.</p> <p>Endian swap size Description</p> <p>b000 No swap, 8-bit data</p> <p>b001 Swap bytes within 16-bit data</p> <p>b010 Swap bytes within 32-bit data</p> <p>b011 Swap bytes within 64-bit data</p> <p>b100 Swap bytes within 128-bit data</p> <p>b101 Reserved</p> <p>b110 Reserved</p> <p>b111 Reserved</p>
dst_cache_ctrl	27:25	sro,ns sraz,n snsro	0x0	<p>Programs the state of AWCACHE[3,1:0]a when the DMAC writes the destination data.</p> <p>Bit [27] 0 = AWCACHE[3] is LOW 1 = AWCACHE[3] is HIGH.</p> <p>Bit [26] 0 = AWCACHE[1] is LOW 1 = AWCACHE[1] is HIGH.</p> <p>Bit [25] 0 = AWCACHE[0] is LOW 1 = AWCACHE[0] is HIGH.</p> <p>Note</p> <p>AWCACHE[2] is tied LOW by the DMAC.</p> <p>Setting AWCACHE[3,1]=b10 violates the AXI protocol. See the AMBA AXI Protocol Specification.</p>

Field Name	Bits	Type	Reset Value	Description
dst_prot_ctrl	24:22	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of AWPROT[2:0]a when the DMAC writes the destination data.</p> <p>Bit [24] 0 = AWPROT[2] is LOW 1 = AWPROT[2] is HIGH.</p> <p>Bit [23] 0 = AWPROT[1] is LOW 1 = AWPROT[1] is HIGH.</p> <p>Bit [22] 0 = AWPROT[0] is LOW 1 = AWPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program AWPROT[1] LOW, that is, a secure access. If</p> <p>a DMA channel in the Non-secure state attempts to set AWPROT[1] LOW, then the DMA channel aborts</p>
dst_burst_len	21:18	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it writes</p> <p>the destination data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers</p> <p>· · ·</p> <p>b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction</p> <p>is the product of dst_burst_len and dst_burst_size.</p> <p>Note</p> <p>These bits control the state of AWLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
dst_burst_size	17:15	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC writes to the destination:</p> <p>b000 = writes 1 byte per beat b001 = writes 2 bytes per beat b010 = writes 4 bytes per beat b011 = writes 8 bytes per beat b100 = writes 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction is the product of dst_burst_len and dst_burst_size.</p> <p>Note These bits control the state of AWSIZE[2:0].</p>
dst_inc	14	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it writes the destination data:</p> <p>0 = Fixed-address burst. The DMAC signals AWBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals AWBURST[0] HIGH.</p>
src_cache_ctrl	13:11	sro,ns sraz,n snsro	0x0	<p>Set the bits to control the state of ARCACHE[2:0]a when the DMAC reads the source data.</p> <p>Bit [13] 0 = ARCACHE[2] is LOW 1 = ARCACHE[2] is HIGH.</p> <p>Bit [12] 0 = ARCACHE[1] is LOW 1 = ARCACHE[1] is HIGH.</p> <p>Bit [11] 0 = ARCACHE[0] is LOW 1 = ARCACHE[0] is HIGH.</p> <p>Note The DMAC ties ARCACHE[3] LOW. Setting ARCACHE[2:1]=b10 violates the AXI protocol.</p>

Field Name	Bits	Type	Reset Value	Description
src_prot_ctrl	10:8	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of ARPROT[2:0]a when the DMAC reads the source data.</p> <p>Bit [10] 0 = ARPROT[2] is LOW 1 = ARPROT[2] is HIGH.</p> <p>Bit [9] 0 = ARPROT[1] is LOW 1 = ARPROT[1] is HIGH.</p> <p>Bit [8] 0 = ARPROT[0] is LOW 1 = ARPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program ARPROT[1] LOW, that is, a secure access. If a</p> <p>DMA channel in the Non-secure state attempts to set ARPROT[1] LOW, the DMA channel aborts.</p>
src_burst_len	7:4	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it reads</p> <p>the source data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers . . . b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction</p> <p>is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
src_burst_size	3:1	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC reads from the source:</p> <p>b000 = reads 1 byte per beat b001 = reads 2 bytes per beat b010 = reads 4 bytes per beat b011 = reads 8 bytes per beat b100 = reads 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARSIZE[2:0].</p>
src_inc	0	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it reads the source data:</p> <p>0 = Fixed-address burst. The DMAC signals ARBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals ARBURST[0] HIGH.</p>

Register ([dmac](#)) LC0_6

Name	LC0_6
Software Name	XDmaPs_LC0_n_OFFSET(6)
Relative Address	0x000004CC
Absolute Address	dmac0_ns: 0xF80044CC dmac0_s: 0xF80034CC
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 0 for DMA channel 6

Register LC0_6 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter zero for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter zero

Register ([dmac](#)) LC1_6

Name	LC1_6
Software Name	XDmaPs_LC1_n_OFFSET(6)
Relative Address	0x000004D0
Absolute Address	dmac0_ns: 0xF80044D0 dmac0_s: 0xF80034D0
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 1 for DMA channel 6

Register LC1_6 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter one for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter one.

Register ([dmac](#)) SAR7

Name	SAR7
Software Name	XDmaPs_SA_n_OFFSET(7)
Relative Address	0x000004E0
Absolute Address	dmac0_ns: 0xF80044E0 dmac0_s: 0xF80034E0

Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	source address for DMA channel 7

Register SAR7 Details

Field Name	Bits	Type	Reset Value	Description
src_addr	31:0	sro,ns sraz,n snsro	0x0	Address of the source data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) DAR7

Name	DAR7
Software Name	XDmaPs_DA_n_OFFSET(7)
Relative Address	0x000004E4
Absolute Address	dmac0_ns: 0xF80044E4 dmac0_s: 0xF80034E4
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	destination address for DMA channel 7

Register DAR7 Details

Field Name	Bits	Type	Reset Value	Description
dest_addr	31:0	sro,ns sraz,n snsro	0x0	Address for the destination data for DMA channel n, where n depends on the address of the register.

Register ([dmac](#)) CCR7

Name	CCR7
Software Name	XDmaPs_CC_n_OFFSET(7)
Relative Address	0x000004E8
Absolute Address	dmac0_ns: 0xF80044E8 dmac0_s: 0xF80034E8
Width	32 bits
Access Type	mixed

Reset Value dmac0_ns: 0x00000000
 dmac0_s: 0x00800200

Description channel control for DMA channel 7

Register CCR7 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	rud	0x0	reserved, read undefined
endian_swap_size	30:28	sro,ns sraz,n snsro	0x0	<p>Table 3-22 on page 3-29 defines whether data can be swapped between little-endian (LE) and byte-invariant big-endian (BE-8) formats, and if so, also defines the natural width of the data independently of the source and destination transaction sizes. This enables unaligned data streams to use the full bus-width, and to be correctly transformed, irrespective of the source and destination address alignments. The format is identical to AxsIZE, except that b000 indicates that no swap must occur.</p> <p>Endian swap size Description</p> <p>b000 No swap, 8-bit data</p> <p>b001 Swap bytes within 16-bit data</p> <p>b010 Swap bytes within 32-bit data</p> <p>b011 Swap bytes within 64-bit data</p> <p>b100 Swap bytes within 128-bit data</p> <p>b101 Reserved</p> <p>b110 Reserved</p> <p>b111 Reserved</p>
dst_cache_ctrl	27:25	sro,ns sraz,n snsro	0x0	<p>Programs the state of AWCACHE[3,1:0]a when the DMAC writes the destination data.</p> <p>Bit [27] 0 = AWCACHE[3] is LOW 1 = AWCACHE[3] is HIGH.</p> <p>Bit [26] 0 = AWCACHE[1] is LOW 1 = AWCACHE[1] is HIGH.</p> <p>Bit [25] 0 = AWCACHE[0] is LOW 1 = AWCACHE[0] is HIGH.</p> <p>Note</p> <p>AWCACHE[2] is tied LOW by the DMAC.</p> <p>Setting AWCACHE[3,1]=b10 violates the AXI protocol. See the AMBA AXI Protocol Specification.</p>

Field Name	Bits	Type	Reset Value	Description
dst_prot_ctrl	24:22	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of AWPROT[2:0]a when the DMAC writes the destination data.</p> <p>Bit [24] 0 = AWPROT[2] is LOW 1 = AWPROT[2] is HIGH.</p> <p>Bit [23] 0 = AWPROT[1] is LOW 1 = AWPROT[1] is HIGH.</p> <p>Bit [22] 0 = AWPROT[0] is LOW 1 = AWPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program AWPROT[1] LOW, that is, a secure access. If</p> <p>a DMA channel in the Non-secure state attempts to set AWPROT[1] LOW, then the DMA channel aborts</p>
dst_burst_len	21:18	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it writes</p> <p>the destination data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers</p> <p>· · ·</p> <p>b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction</p> <p>is the product of dst_burst_len and dst_burst_size.</p> <p>Note</p> <p>These bits control the state of AWLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
dst_burst_size	17:15	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC writes to the destination:</p> <p>b000 = writes 1 byte per beat b001 = writes 2 bytes per beat b010 = writes 4 bytes per beat b011 = writes 8 bytes per beat b100 = writes 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction</p> <p>is the product of dst_burst_len and dst_burst_size.</p> <p>Note</p> <p>These bits control the state of AWSIZE[2:0].</p>
dst_inc	14	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it writes the destination data:</p> <p>0 = Fixed-address burst. The DMAC signals AWBURST[0] LOW.</p> <p>1 = Incrementing-address burst. The DMAC signals AWBURST[0] HIGH.</p>
src_cache_ctrl	13:11	sro,ns sraz,n snsro	0x0	<p>Set the bits to control the state of ARCACHE[2:0]a when the DMAC reads the source data.</p> <p>Bit [13] 0 = ARCACHE[2] is LOW 1 = ARCACHE[2] is HIGH.</p> <p>Bit [12] 0 = ARCACHE[1] is LOW 1 = ARCACHE[1] is HIGH.</p> <p>Bit [11] 0 = ARCACHE[0] is LOW 1 = ARCACHE[0] is HIGH.</p> <p>Note</p> <p>The DMAC ties ARCACHE[3] LOW.</p> <p>Setting ARCACHE[2:1]=b10 violates the AXI protocol.</p>

Field Name	Bits	Type	Reset Value	Description
src_prot_ctrl	10:8	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	<p>Programs the state of ARPROT[2:0]a when the DMAC reads the source data.</p> <p>Bit [10] 0 = ARPROT[2] is LOW 1 = ARPROT[2] is HIGH.</p> <p>Bit [9] 0 = ARPROT[1] is LOW 1 = ARPROT[1] is HIGH.</p> <p>Bit [8] 0 = ARPROT[0] is LOW 1 = ARPROT[0] is HIGH.</p> <p>Note</p> <p>Only DMA channels in the Secure state can program ARPROT[1] LOW, that is, a secure access. If a</p> <p>DMA channel in the Non-secure state attempts to set ARPROT[1] LOW, the DMA channel aborts.</p>
src_burst_len	7:4	sro,ns sraz,n snsro	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it reads</p> <p>the source data:</p> <p>b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers . . . b1111 = 16 data transfers.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction</p> <p>is the product of src_burst_len and src_burst_size.</p> <p>Note</p> <p>These bits control the state of ARLEN[3:0].</p>

Field Name	Bits	Type	Reset Value	Description
src_burst_size	3:1	sro,ns sraz,n snsro	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC reads from the source:</p> <p>b000 = reads 1 byte per beat b001 = reads 2 bytes per beat b010 = reads 4 bytes per beat b011 = reads 8 bytes per beat b100 = reads 16 bytes per beat b101-b111 = reserved.</p> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size.</p> <p>Note These bits control the state of ARSIZE[2:0].</p>
src_inc	0	sro,ns sraz,n snsro	0x0	<p>Programs the burst type that the DMAC performs when it reads the source data:</p> <p>0 = Fixed-address burst. The DMAC signals ARBURST[0] LOW. 1 = Incrementing-address burst. The DMAC signals ARBURST[0] HIGH.</p>

Register ([dmac](#)) LC0_7

Name	LC0_7
Software Name	XDmaPs_LC0_n_OFFSET(7)
Relative Address	0x000004EC
Absolute Address	dmac0_ns: 0xF80044EC dmac0_s: 0xF80034EC
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 0 for DMA channel 7

Register LC0_7 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter zero for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter zero

Register ([dmac](#)) LC1_7

Name	LC1_7
Software Name	XDmaPs_LC1_n_OFFSET(7)
Relative Address	0x000004F0
Absolute Address	dmac0_ns: 0xF80044F0 dmac0_s: 0xF80034F0
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	loop counter 1 for DMA channel 7

Register LC1_7 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	reserved, read undefined
loop_counter_iteration	7:0	sro,ns sraz,n snsro	0x0	Provides the status of loop counter one for the DMA channel. The DMAC updates this register when it executes DMALPEND[S B], and the DMA channel thread is programmed to use loop counter one.

Register ([dmac](#)) DBGSTATUS

Name	DBGSTATUS
Relative Address	0x00000D00
Absolute Address	dmac0_ns: 0xF8004D00 dmac0_s: 0xF8003D00
Width	32 bits

Access Type mixed

Reset Value 0x00000000

Description Debug Status Register

Register DBGSTATUS Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rud	0x0	reserved, read undefined
dbgstatus	0	sro,ns sraz,n snsro	0x0	The debug status encoding is: 0 = Idle 1 = Busy.

Register ([dmac](#)) DBGCMD

Name DBGCMD

Relative Address 0x00000D04

Absolute Address dmac0_ns: 0xF8004D04
dmac0_s: 0xF8003D04

Width 32 bits

Access Type mixed

Reset Value 0x00000000

Description Debug Command Register

Register DBGCMD Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:2	rud	0x0	reserved, read undefined
dbgcmd	1:0	swo,n ssraz, nsns wo	0x0	The debug encoding is as follows: b00 = execute the instruction that the DBGINST [1:0] Registers contain b01 = reserved b10 = reserved b11 = reserved.

Register ([dmac](#)) DBGINST0

Name DBGINST0

Relative Address 0x00000D08

Absolute Address dmac0_ns: 0xF8004D08
dmac0_s: 0xF8003D08

Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	debug instruction 0 register, Controls the debug instruction, channel, and thread information for the DMAC.

Register DBGINST0 Details

Field Name	Bits	Type	Reset Value	Description
instruction_byte1	31:24	swo,n ssraz, nsns wo	0x0	instruction byte 1
instruction_byte0	23:16	swo,n ssraz, nsns wo	0x0	instruction byte 0
reserved	15:11	waz	0x0	reserved, write as 0
channel_num	10:8	swo,n ssraz, nsns wo	0x0	DMA channel number: b000 = DMA channel 0 b001 = DMA channel 1 b010 = DMA channel 2 . . . b111 = DMA channel 7.
reserved	7:1	waz	0x0	reserved, write as 0
debug_thread	0	swo,n ssraz, nsns wo	0x0	The debug thread encoding is as follows: 0 = DMA manager thread 1 = DMA channel. Note When set to 1, the Channel number field selects the DMA channel to debug.

Register ([dmac](#)) DBGINST1

Name	DBGINST1
Relative Address	0x0000D0C
Absolute Address	dmac0_ns: 0xF8004D0C dmac0_s: 0xF8003D0C
Width	32 bits

Access Type	mixed
Reset Value	0x00000000
Description	debug instruction 0 register, Controls the upper bytes of the debug instruction for the DMAC

Register DBGINST1 Details

Field Name	Bits	Type	Reset Value	Description
instruction_byte5	31:24	swo,n ssraz, nsns wo	0x0	instruction byte 5
instruction_byte4	23:16	swo,n ssraz, nsns wo	0x0	instruction byte 4
instruction_byte3	15:8	swo,n ssraz, nsns wo	0x0	instruction byte 3
instruction_byte2	7:0	swo,n ssraz, nsns wo	0x0	instruction byte 2

Register ([dmac](#)) CR0

Name	CR0
Relative Address	0x00000E00
Absolute Address	dmac0_ns: 0xF8004E00 dmac0_s: 0xF8003E00
Width	32 bits
Access Type	mixed
Reset Value	dmac0_ns: 0x00000000 dmac0_s: 0x001E3071
Description	Configuration Register 0, Provides the status of the tie-off control signals.

Register CR0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rud	0x0	read undefined
num_events	21:17	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0xF	Number of interrupt outputs that the DMAC provides: b00000 = 1 interrupt output, irq[0] b00001 = 2 interrupt outputs, irq[1:0] b00010 = 3 interrupt outputs, irq[2:0] . . . b11111 = 32 interrupt outputs, irq[31:0].
num_periph_req	16:12	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x3	Number of peripheral request interfaces that the DMAC provides: b00000 = 1 peripheral request interface b00001 = 2 peripheral request interfaces b00010 = 3 peripheral request interfaces . . . b11111 = 32 peripheral request interfaces. Note This field is only valid when the periph_req bit is set to 1.
reserved	11:7	rud	0x0	read undefined
num_chnls	6:4	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x7	Number of DMA channels that the DMAC supports: b000 = 1 DMA channel b001 = 2 DMA channels b010 = 3 DMA channels . . . b111 = 8 DMA channels.
reserved	3	rud	0x0	read undefined
mgr_ns_at_rst	2	sro,ns sraz,n snsro	0x0	Indicates the status of the boot_manager_ns signal when the DMAC exited from reset: 0 = boot_manager_ns was LOW 1 = boot_manager_ns was HIGH.

Field Name	Bits	Type	Reset Value	Description
boot_en	1	sro,ns sraz,n snsro	0x0	Indicates the status of the boot_from_pc signal when the DMAC exited from reset: 0 = boot_from_pc was LOW 1 = boot_from_pc was HIGH.
periph_req	0	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x1	Supports peripheral requests: 0 = the DMAC does not provide a peripheral request interface 1 = the DMAC provides the number of peripheral request interfaces that the num_periph_req field specifies.

Register ([dmac](#)) CR1

Name	CR1
Relative Address	0x00000E04
Absolute Address	dmac0_ns: 0xF8004E04 dmac0_s: 0xF8003E04
Width	32 bits
Access Type	mixed
Reset Value	dmac0_ns: 0x00000000 dmac0_s: 0x00000074
Description	Configuration Register 1, Provides information about the instruction cache configuration.

Register CR1 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	read undefined
num_icache_lines	7:4	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x7	Number of i-cache lines: b0000 = 1 i-cache line b0001 = 2 i-cache lines b0010 = 3 i-cache lines . . . b1111 = 16 i-cache lines

Field Name	Bits	Type	Reset Value	Description
reserved	3	rud	0x0	read undefined
icache_len	2:0	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x4	The length of an i-cache line: b000-b001 = reserved b010 = 4 bytes b011 = 8 bytes b100 = 16 bytes b101 = 32 bytes b110-b111 = reserved.

Register ([dmac](#)) CR2

Name	CR2
Relative Address	0x00000E08
Absolute Address	dmac0_ns: 0xF8004E08 dmac0_s: 0xF8003E08
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Configuration Register 2, Provides the value of the boot address that boot_addr[31:0] configures.

Register CR2 Details

Field Name	Bits	Type	Reset Value	Description
boot_addr	31:0	sro,ns sraz,n snsro	0x0	Provides the value of boot_addr[31:0] when the DMAC exited from reset

Register ([dmac](#)) CR3

Name	CR3
Relative Address	0x00000E0C
Absolute Address	dmac0_ns: 0xF8004E0C dmac0_s: 0xF8003E0C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000

Description Configuration Register 3, Provides the security state of the event-interrupt resources that are initialized when the DMAC exits from reset.

Register CR3 Details

Field Name	Bits	Type	Reset Value	Description
INS	31:0	sro,ns sraz,n snsro	0x0	Provides the security state of an event-interrupt resource: Bit [N] = 0 Event<N> or irq[N] is in the Secure state. Bit [N] = 1 Event<N> or irq[N] is in the Non-secure state. Note The boot_irq_ns[x:0] signals initialize the bits in this register when the DMAC exits from reset.

Register ([dmac](#)) CR4

Name CR4

Relative Address 0x00000E10

Absolute Address dmac0_ns: 0xF8004E10
dmac0_s: 0xF8003E10

Width 32 bits

Access Type mixed

Reset Value 0x00000000

Description Configuration Register 4, Provides the security state of the peripheral request interfaces that is initialized when the DMAC exits from reset.

Register CR4 Details

Field Name	Bits	Type	Reset Value	Description
PNS	31:0	sro,ns sraz,n snsro	0x0	Provides the security state of the peripheral request interfaces: Bit [N] = 0 Peripheral request interface N is in the Secure state. Bit [N] = 1 Peripheral request interface N is in the Non-secure state. Note The boot_periph_ns tie-off signals initialize the bits in this register when the DMAC exits from reset. See Table A-12 on page A-9 for more information.

Register ([dmac](#)) CRD

Name	CRD
Software Name	CRDN
Relative Address	0x00000E14
Absolute Address	dmac0_ns: 0xF8004E14 dmac0_s: 0xF8003E14
Width	32 bits
Access Type	mixed
Reset Value	dmac0_ns: 0x00000000 dmac0_s: 0x07FF7F73
Description	DMA configuration register, Provides information about the configuration of the data buffer, data width, and read and write issuing capability of the DMAC.

Register CRD Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:30	rud	0x0	read undefined
data_buffer_dep	29:20	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x7F	The number of lines that the data buffer contains: b000000000 = 1 line b000000001 = 2 lines . . . b111111111 = 1024 lines.
rd_q_dep	19:16	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0xF	The depth of the read queue: b0000 = 1 line b0001 = 2 lines . . . b1111 = 16 lines.
reserved	15	rud	0x0	read undefined
rd_cap	14:12	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x7	Read issuing capability that programs the number of outstanding read transactions: b000 = 1 b001 = 2 . . . b111 = 8.

Field Name	Bits	Type	Reset Value	Description
wr_q_dep	11:8	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0xF	The depth of the write queue: b0000 = 1 line b0001 = 2 lines . . . b1111 = 16 lines.
reserved	7	rud	0x0	read undefined
wr_cap	6:4	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x7	Write issuing capability that programs the number of outstanding write transactions: b000 = 1 b001 = 2 . . . b111 = 8.
reserved	3	rud	0x0	read undefined
data_width	2:0	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x3	The data bus width of the AXI master interface: b000 = reserved b001 = reserved b010 = 32-bit b011 = 64-bit b100 = 128-bit b101-b111 = reserved.

Register ([dmac](#)) WD

Name	WD
Relative Address	0x00000E80
Absolute Address	dmac0_ns: 0xF8004E80 dmac0_s: 0xF8003E80
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	watch dog register, control the watch dog behavior

Register WD Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rud	0x0	read undefined
wd_irq_only	0	sro,ns sraz,n snsro	0x0	Controls how the DMAC responds when it detects a lock-up condition: 0 = the DMAC aborts all of the contributing DMA channels and sets irq_abort HIGH 1 = the DMAC sets irq_abort HIGH.

Register ([dmac](#)) periph_id_0

Name	periph_id_0
Software Name	PERIPH_ID_0
Relative Address	0x00000FE0
Absolute Address	dmac0_ns: 0xF8004FE0 dmac0_s: 0xF8003FE0
Width	32 bits
Access Type	mixed
Reset Value	dmac0_ns: 0x00000000 dmac0_s: 0x00000030
Description	peripheral identification register 0, Provides information about the configuration and version of the peripheral

Register periph_id_0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	read undefined
part_number_0	7:0	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x30	returns 0x30

Register ([dmac](#)) periph_id_1

Name	periph_id_1
Software Name	PERIPH_ID_1
Relative Address	0x00000FE4
Absolute Address	dmac0_ns: 0xF8004FE4 dmac0_s: 0xF8003FE4
Width	32 bits
Access Type	mixed

Reset Value	dmac0_ns: 0x00000000 dmac0_s: 0x00000013
Description	peripheral identification register 1, The periph_id_1 Register is hard-coded and the fields in the register control the reset value.

Register periph_id_1 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	read undefined
designer_0	7:4	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x1	returns 0x1
part_number_1	3:0	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x3	returns 0x3

Register ([dmac](#)) periph_id_2

Name	periph_id_2
Software Name	PERIPH_ID_2
Relative Address	0x00000FE8
Absolute Address	dmac0_ns: 0xF8004FE8 dmac0_s: 0xF8003FE8
Width	32 bits
Access Type	mixed
Reset Value	dmac0_ns: 0x00000000 dmac0_s: 0x00000024
Description	peripheral identification register 2, The periph_id_2 Register is hard-coded and the fields in the register control the reset value.

Register periph_id_2 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	read undefined
revision	7:4	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x2	Identifies the revision: 0x0 for r0p0 0x1 for r1p0 0x2 for r1p1.
designer_1	3:0	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x4	returns 0x4

Register ([dmac](#)) periph_id_3

Name	periph_id_3
Software Name	PERIPH_ID_3
Relative Address	0x00000FEC
Absolute Address	dmac0_ns: 0xF8004FEC dmac0_s: 0xF8003FEC
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	peripheral identification register 3, Provides information about the configuration and version of the peripheral

Register periph_id_3 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rud	0x0	read undefined
integration_cfg	0	sro,ns sraz,n snsro	0x0	Returns 0 to indicate that the DMAC does not contain integration test logic

Register ([dmac](#)) pcell_id_0

Name	pcell_id_0
Software Name	PCELL_ID_0
Relative Address	0x00000FF0
Absolute Address	dmac0_ns: 0xF8004FF0 dmac0_s: 0xF8003FF0
Width	32 bits
Access Type	mixed
Reset Value	dmac0_ns: 0x00000000 dmac0_s: 0x0000000D
Description	component identification register 0, When concatenated, these four registers return 0xB105F00D.

Register pcell_id_0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	read undefined
pcell_id_0	7:0	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0xD	return 0x0d

Register ([dmac](#)) pcell_id_1

Name	pcell_id_1
Software Name	PCELL_ID_1
Relative Address	0x00000FF4
Absolute Address	dmac0_ns: 0xF8004FF4 dmac0_s: 0xF8003FF4
Width	32 bits
Access Type	mixed
Reset Value	dmac0_ns: 0x00000000 dmac0_s: 0x000000F0
Description	component identification register 0, When concatenated, these four registers return 0xB105F00D.

Register pcell_id_1 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	read undefined
pcell_id_1	7:0	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0xF0	returns 0xf0

Register ([dmac](#)) pcell_id_2

Name	pcell_id_2
Software Name	PCELL_ID_2
Relative Address	0x00000FF8
Absolute Address	dmac0_ns: 0xF8004FF8 dmac0_s: 0xF8003FF8
Width	32 bits
Access Type	mixed

Reset Value	dmac0_ns: 0x00000000 dmac0_s: 0x00000005
Description	component identification register 0, When concatenated, these four registers return 0xB105F00D.

Register pcell_id_2 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	read undefined
pcell_id_2	7:0	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0x5	returns 0x05

Register ([dmac](#)) pcell_id_3

Name	pcell_id_3
Software Name	PCELL_ID_3
Relative Address	0x00000FFC
Absolute Address	dmac0_ns: 0xF8004FFC dmac0_s: 0xF8003FFC
Width	32 bits
Access Type	mixed
Reset Value	dmac0_ns: 0x00000000 dmac0_s: 0x000000B1
Description	component identification register 0, When concatenated, these four registers return 0xB105F00D.

Register pcell_id_3 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rud	0x0	read undefined
pcell_id_3	7:0	sro,ns sraz,n snsro	dmac0_ns: 0x0 dmac0_s: 0xB1	returns 0xb1

B.18 Gigabit Ethernet Controller (GEM)

Module Name	Gigabit Ethernet Controller (GEM)
Base Address	0xE000B000 gem0 0xE000C000 gem1
Description	Gigabit Ethernet Controller Instance no. 0.
Version	1.0
Doc Version	
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
net_ctrl	0x00000000	32	mixed	0x00000000	Network Control
net_cfg	0x00000004	32	rw	0x00080000	Network Configuration
net_status	0x00000008	32	ro	x	Network Status
user_io	0x0000000C	32	mixed	x	User Input/Output
dma_cfg	0x00000010	32	mixed	0x00020784	DMA Configuration
tx_status	0x00000014	32	mixed	0x00000000	Transmit Status
rx_qbar	0x00000018	32	mixed	0x00000000	Receive Buffer Queue Base Address
tx_qbar	0x0000001C	32	mixed	0x00000000	Transmit Buffer Queue Base Address
rx_status	0x00000020	32	mixed	0x00000000	Receive Status
intr_status	0x00000024	32	ro	0x00000000	Interrupt Status, normally read-only
intr_en	0x00000028	32	wo	x	Interrupt Enable, write-only
intr_dis	0x0000002C	32	wo	x	Interrupt Disable, write-only
intr_mask	0x00000030	32	mixed	x	Interrupt Mask Status, normally read-only
phy_maint	0x00000034	32	rw	0x00000000	PHY Maintenance
rx_pauseq	0x00000038	32	ro	0x00000000	Received Pause Quantum
tx_pauseq	0x0000003C	32	rw	0x0000FFFF	Transmit Pause Quantum
hash_bot	0x00000080	32	rw	0x00000000	Hash Register Bottom [31:0]
hash_top	0x00000084	32	rw	0x00000000	Hash Register Top [63:32]

Register Name	Address	Width	Type	Reset Value	Description
spec_addr1_bot	0x00000088	32	rw	0x00000000	Specific Address 1 Bottom [31:0]
spec_addr1_top	0x0000008C	32	mixed	0x00000000	Specific Address 1 Top [47:32]
spec_addr2_bot	0x00000090	32	rw	0x00000000	Specific Address 2 Bottom [31:0]
spec_addr2_top	0x00000094	32	mixed	0x00000000	Specific Address 2 Top [47:32]
spec_addr3_bot	0x00000098	32	rw	0x00000000	Specific Address 3 Bottom [31:0]
spec_addr3_top	0x0000009C	32	mixed	0x00000000	Specific Address 3 Top [47:32]
spec_addr4_bot	0x000000A0	32	rw	0x00000000	Specific Address 4 Bottom [31:0]
spec_addr4_top	0x000000A4	32	mixed	0x00000000	Specific Address 4 Top [47:32]
type_id_match1	0x000000A8	32	mixed	0x00000000	Type ID Match 1
type_id_match2	0x000000AC	32	mixed	0x00000000	Type ID Match 2
type_id_match3	0x000000B0	32	mixed	0x00000000	Type ID Match 3
type_id_match4	0x000000B4	32	mixed	0x00000000	Type ID Match 4
wake_on_lan	0x000000B8	32	mixed	0x00000000	Wake on LAN Register
ipg_stretch	0x000000BC	32	mixed	0x00000000	IPG stretch register
stacked_vlan	0x000000C0	32	mixed	0x00000000	Stacked VLAN Register
tx_pfc_pause	0x000000C4	32	mixed	0x00000000	Transmit PFC Pause Register
spec_addr1_mask_bot	0x000000C8	32	rw	0x00000000	Specific Address Mask 1 Bottom [31:0]
spec_addr1_mask_top	0x000000CC	32	mixed	0x00000000	Specific Address Mask 1 Top [47:32]
module_id	0x000000FC	32	ro	0x00020118	Module ID
octets_tx_bot	0x00000100	32	ro	0x00000000	Octets transmitted [31:0] (in frames without error)
octets_tx_top	0x00000104	32	ro	0x00000000	Octets transmitted [47:32] (in frames without error)
frames_tx	0x00000108	32	ro	0x00000000	Frames Transmitted, normally read-only
broadcast_frames_tx	0x0000010C	32	ro	0x00000000	Broadcast frames Tx, normally read-only
multi_frames_tx	0x00000110	32	ro	0x00000000	Multicast frames Tx, normally read-only
pause_frames_tx	0x00000114	32	ro	0x00000000	Pause frames Tx, normally read-only
frames_64b_tx	0x00000118	32	ro	0x00000000	Frames Tx, 64-byte length
frames_65to127b_tx	0x0000011C	32	ro	0x00000000	Frames Tx, 65 to 127-byte length
frames_128to255b_tx	0x00000120	32	ro	0x00000000	Frames Tx, 128 to 255-byte length

Register Name	Address	Width	Type	Reset Value	Description
frames_256to511b_tx	0x00000124	32	ro	0x00000000	Frames Tx, 256 to 511-byte length
frames_512to1023b_tx	0x00000128	32	ro	0x00000000	Frames Tx, 512 to 1023-byte length
frames_1024to1518b_tx	0x0000012C	32	ro	0x00000000	Frame Tx, 1024 to 1518-byte length
tx_under_runs	0x00000134	32	ro	0x00000000	Transmit under runs
single_collisn_frames	0x00000138	32	ro	0x00000000	Single Collision Frames
multi_collisn_frames	0x0000013C	32	ro	0x00000000	Multiple Collision Frames
excessive_collisns	0x00000140	32	ro	0x00000000	Excessive Collisions
late_collisns	0x00000144	32	ro	0x00000000	Late Collisions
deferred_tx_frames	0x00000148	32	ro	0x00000000	Deferred Transmission Frames
carrier_sense_errs	0x0000014C	32	ro	0x00000000	Carrier Sense Errors.
octets_rx_bot	0x00000150	32	ro	0x00000000	Octets Received [31:0]
octets_rx_top	0x00000154	32	ro	0x00000000	Octets Received [47:32]
frames_rx	0x00000158	32	ro	0x00000000	Frames Received, normally read-only
bdcast_fames_rx	0x0000015C	32	ro	0x00000000	Broadcast Frames Rx
multi_frames_rx	0x00000160	32	ro	0x00000000	Multicast Frames Rx
pause_rx	0x00000164	32	ro	0x00000000	Pause Frames Rx
frames_64b_rx	0x00000168	32	ro	0x00000000	Frames Rx, 64-byte length
frames_65to127b_rx	0x0000016C	32	ro	0x00000000	Frames Rx, 65 to 127-byte length
frames_128to255b_rx	0x00000170	32	ro	0x00000000	Frames Rx, 128 to 255-byte length
frames_256to511b_rx	0x00000174	32	ro	0x00000000	Frames Rx, 256 to 511-byte length
frames_512to1023b_rx	0x00000178	32	ro	0x00000000	Frames Rx, 512 to 1023-byte length
frames_1024to1518b_rx	0x0000017C	32	ro	0x00000000	Frames Rx, 1024 to 1518-byte length
undersz_rx	0x00000184	32	ro	0x00000000	Undersize frames received
oversz_rx	0x00000188	32	ro	0x00000000	Oversize frames received
jab_rx	0x0000018C	32	ro	0x00000000	Jabbers received
fcs_errors	0x00000190	32	ro	0x00000000	Frame check sequence errors
length_field_errors	0x00000194	32	ro	0x00000000	Length field frame errors
rx_symbol_errors	0x00000198	32	ro	0x00000000	Receive symbol errors
align_errors	0x0000019C	32	ro	0x00000000	Alignment errors

Register Name	Address	Width	Type	Reset Value	Description
rx_resource_errors	0x000001A0	32	ro	0x00000000	Receive resource errors
rx_overrun_errors	0x000001A4	32	ro	0x00000000	Receive overrun errors
ip_hdr_csum_errors	0x000001A8	32	ro	0x00000000	IP header checksum errors
tcp_csum_errors	0x000001AC	32	ro	0x00000000	TCP checksum errors
udp_csum_errors	0x000001B0	32	ro	0x00000000	UDP checksum error
timer_strobe_s	0x000001C8	32	rw	0x00000000	1588 timer sync strobe seconds
timer_strobe_ns	0x000001CC	32	mixed	0x00000000	1588 timer sync strobe nanoseconds
timer_s	0x000001D0	32	rw	0x00000000	1588 timer seconds
timer_ns	0x000001D4	32	mixed	0x00000000	1588 timer nanoseconds
timer_adjust	0x000001D8	32	mixed	0x00000000	1588 timer adjust
timer_incr	0x000001DC	32	mixed	0x00000000	1588 timer increment
ptp_tx_s	0x000001E0	32	ro	0x00000000	PTP event frame transmitted seconds
ptp_tx_ns	0x000001E4	32	ro	0x00000000	PTP event frame transmitted nanoseconds
ptp_rx_s	0x000001E8	32	ro	0x00000000	PTP event frame received seconds
ptp_rx_ns	0x000001EC	32	ro	0x00000000	PTP event frame received nanoseconds.
ptp_peer_tx_s	0x000001F0	32	ro	0x00000000	PTP peer event frame transmitted seconds
ptp_peer_tx_ns	0x000001F4	32	ro	0x00000000	PTP peer event frame transmitted nanoseconds
ptp_peer_rx_s	0x000001F8	32	ro	0x00000000	PTP peer event frame received seconds
ptp_peer_rx_ns	0x000001FC	32	ro	0x00000000	PTP peer event frame received nanoseconds.
design_cfg2	0x00000284	32	ro	x	Design Configuration 2
design_cfg3	0x00000288	32	ro	0x00000000	Design Configuration 3
design_cfg4	0x0000028C	32	ro	0x00000000	Design Configuration 4
design_cfg5	0x00000290	32	ro	x	Design Configuration 5

Register ([GEM](#)) net_ctrl

Name net_ctrl
Software Name XEMACPS_NWCTRL

Relative Address	0x00000000
Absolute Address	gem0: 0xE000B000 gem1: 0xE000C000
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Network Control

Register net_ctrl Details

The network control register contains general MAC control functions for both receiver and transmitter.

Field Name	Bits	Type	Reset Value	Description
reserved	31:19	ro	0x0	Reserved, read as zero, ignored on write.
flush_next_rx_dpram_pkt	18	wo	0x0	Flush the next packet from the external RX DPRAM. Writing one to this bit will only have an effect if the DMA is not currently writing a packet already stored in the DPRAM to memory.
tx_pfc_pri_pause_frame	17	wo	0x0	Transmit PFC Priority Based Pause Frame. Takes the values stored in the Transmit PFC Pause Register
en_pfc_pri_pause_rx	16	wo	0x0	Enable PFC Priority Based Pause Reception capabilities. Setting this bit will enable PFC negotiation and recognition of priority based pause frames.
str_rx_timestamp	15	rw	0x0	Store receive time stamp to memory. Setting this bit to one will cause the CRC of every received frame to be replaced with the value of the nanoseconds field of the 1588 timer that was captured as the receive frame passed the message time stamp point. Set to zero for normal operation.
reserved	14	rw	0x0	Reserved. Do not modify.
reserved	13	wo	0x0	Reserved. Do not modify.
tx_zeroq_pause_frame (ZEROPAUSCTX)	12	wo	0x0	Transmit zero quantum pause frame - writing one to this bit causes a pause frame with zero quantum to be transmitted.
tx_pause_frame (PAUSCTX)	11	wo	0x0	Transmit pause frame - writing one to this bit causes a pause frame to be transmitted.
tx_halt (HALTTX)	10	wo	0x0	Transmit halt - writing one to this bit halts transmission as soon as any ongoing frame transmission ends.

Field Name	Bits	Type	Reset Value	Description
start_tx (STARTTX)	9	wo	0x0	Start transmission - writing one to this bit starts transmission.
back_pressure	8	rw	0x0	Back pressure - if set in 10M or 100M half duplex mode will force collisions on all received frames. Ignored in gigabit half duplex mode.
wren_stat_regs (STATWEN)	7	rw	0x0	Write enable for statistics registers - setting this bit to one means the statistics registers can be written for functional test purposes.
incr_stat_regs (STATINC)	6	wo	0x0	Incremental statistics registers - this bit is write only. Writing a one increments all the statistics registers by one for test purposes.
clear_stat_regs (STATCLR)	5	wo	0x0	Clear statistics registers - this bit is write only. Writing a one clears the statistics registers.
mgmt_port_en (MDEN)	4	rw	0x0	Management port enable - set to one to enable the management port. When zero forces mdio to high impedance state and mdc low.
tx_en (TXEN)	3	rw	0x0	Transmit enable - when set, it enables the GEM transmitter to send data. When reset transmission will stop immediately, the transmit pipeline and control registers will be cleared and the transmit queue pointer register will reset to point to the start of the transmit descriptor list.
rx_en (RXEN)	2	rw	0x0	Receive enable - when set, it enables the GEM to receive data. When reset frame reception will stop immediately and the receive pipeline will be cleared. The receive queue pointer register is unaffected.
loopback_local (LOOPEN)	1	rw	0x0	Loop back local - asserts the loopback_local signal to the system clock generator. Also connects txd to rxd, tx_en to rx_dv and forces full duplex mode. Bit 11 of the network configuration register must be set low to disable TBI mode when in internal loopback. rx_clk and tx_clk may malfunction as the GEM is switched into and out of internal loop back. It is important that receive and transmit circuits have already been disabled when making the switch into and out of internal loop back. Local loopback functionality isn't available in the EP107 Zynq Emulation Platform, because the clocking doesn't map well into an FPGA.
reserved	0	rw	0x0	Reserved. Do not modify.

Register (GEM) net_cfg

Name	net_cfg
Software Name	XEMACPS_NWCFG
Relative Address	0x00000004
Absolute Address	gem0: 0xE000B004 gem1: 0xE000C004
Width	32 bits
Access Type	rw
Reset Value	0x00080000
Description	Network Configuration

Register net_cfg Details

The network configuration register contains functions for setting the mode of operation for the Gigabit Ethernet MAC

Field Name	Bits	Type	Reset Value	Description
unidir_en	31	rw	0x0	Uni-direction-enable. When low the PCS will transmit idle symbols if the link goes down. When high the PCS can transmit frame data when the link is down.
ignore_ipg_rx_er	30	rw	0x0	Ignore IPG rx_er. When set rx_er has no effect on the GEM's operation when rx_dv is low. Set this when using the RGMII wrapper in half-duplex mode.
rx_bad_preamble (BADPREAMBEN)	29	rw	0x0	Receive bad preamble. When set frames with non-standard preamble are not rejected.
ipg_stretch_en (IPDSTRETCH)	28	rw	0x0	IPG stretch enable - when set the transmit IPG can be increased above 96 bit times depending on the previous frame length using the IPG stretch register.
sgmii_en	27	rw	0x0	SGMII mode enable - changes behavior of the auto-negotiation advertisement and link partner ability registers to meet the requirements of SGMII and reduces the duration of the link timer from 10 ms to 1.6 ms
ignore_rx_fcs (FCSIGNORE)	26	rw	0x0	Ignore RX FCS - when set frames with FCS/CRC errors will not be rejected. FCS error statistics will still be collected for frames with bad FCS and FCS status will be recorded in frame's DMA descriptor. For normal operation this bit must be set to zero.
rx_hd_while_tx (HDRXEN)	25	rw	0x0	Enable frames to be received in half-duplex mode while transmitting.

Field Name	Bits	Type	Reset Value	Description
rx_chksum_offld_en (RXCHKSUMEN)	24	rw	0x0	Receive checksum offload enable - when set, the receive checksum engine is enabled. Frames with bad IP, TCP or UDP checksums are discarded.
dis_cp_pause_frame (PAUSECOPYDI)	23	rw	0x0	Disable copy of pause frames - set to one to prevent valid pause frames being copied to memory. When set, pause frames are not copied to memory regardless of the state of the copy all frames bit; whether a hash match is found or whether a type ID match is identified. If a destination address match is found the pause frame will be copied to memory. Note that valid pause frames received will still increment pause statistics and pause the transmission of frames as required.
dbus_width	22:21	rw	0x0	Data bus width - set according to AMBA AHB or external FIFO data bus width. The reset value for this can be changed by defining a new value for gem_dma_bus_width_def in gem_defs.v. Only valid bus widths may be written if the system is configured to a maximum width less than 128-bits. Zynq defines gem_dma_bus_width_def as 2'b00. 00: 32 bit AMBA AHB data bus width 01: 64 bit AMBA AHB data bus width 10: 128 bit AMBA AHB data bus width 11: 128 bit AMBA AHB data bus width
mdc_clk_div (MDCCLKDIV)	20:18	rw	0x2	MDC clock division - set according to pclk speed. These three bits determine the number pclk will be divided by to generate MDC. For conformance with the 802.3 specification, MDC must not exceed 2.5 MHz (MDC is only active during MDIO read and write operations). The reset value for this can be changed by defining a new value for gem_mdc_clock_div in gem_defs.v. Zynq defines gem_mdc_clock_div as 3'b010. 000: divide pclk by 8 (pclk up to 20 MHz) 001: divide pclk by 16 (pclk up to 40 MHz) 010: divide pclk by 32 (pclk up to 80 MHz) 011: divide pclk by 48 (pclk up to 120MHz) 100: divide pclk by 64 (pclk up to 160 MHz) 101: divide pclk by 96 (pclk up to 240 MHz) 110: divide pclk by 128 (pclk up to 320 MHz) 111: divide pclk by 224 (pclk up to 540 MHz)
fcs_remove (FCSREM)	17	rw	0x0	FCS remove - setting this bit will cause received frames to be written to memory without their frame check sequence (last 4 bytes). The frame length indicated will be reduced by four bytes in this mode.

Field Name	Bits	Type	Reset Value	Description
len_err_frame_disc (LENGTHERRDISCRD)	16	rw	0x0	Length field error frame discard - setting this bit causes frames with a measured length shorter than the extracted length field (as indicated by bytes 13 and 14 in a non-VLAN tagged frame) to be discarded. This only applies to frames with a length field less than 0x0600.
rx_buf_offset (RXOFFS)	15:14	rw	0x0	Receive buffer offset - indicates the number of bytes by which the received data is offset from the start of the receive buffer.
pause_en (PAUSEEN)	13	rw	0x0	Pause enable - when set, transmission will pause if a non zero 802.3 classic pause frame is received and PFC has not been negotiated.
retry_test (RETRYTESTEN)	12	rw	0x0	Retry test - must be set to zero for normal operation. If set to one the backoff between collisions will always be one slot time. Setting this bit to one helps test the too many retries condition. Also used in the pause frame tests to reduce the pause counter's decrement time from 512 bit times, to every rx_clk cycle.
pcs_sel	11	rw	0x0	PCS select - selects between MII/GMII and TBI. Must be set for SGMII operation. 0: GMII/MII interface enabled, TBI disabled 1: TBI enabled, GMII/MII disabled
gige_en (1000)	10	rw	0x0	Gigabit mode enable - setting this bit configures the GEM for 1000 Mbps operation. 0: 10/100 operation using MII or TBI interface 1: Gigabit operation using GMII or TBI interface
ext_addr_match_en (EXTADDRMATCHEN)	9	rw	0x0	External address match enable - when set the external address match interface can be used to copy frames to memory.
reserved	8	rw	0x0	Reserved. Do not modify.
uni_hash_en (UCASTHASHEN)	7	rw	0x0	Unicast hash enable - when set, unicast frames will be accepted when the 6 bit hash function of the destination address points to a bit that is set in the hash register.
multi_hash_en (MCASTHASHEN)	6	rw	0x0	Multicast hash enable - when set, multicast frames will be accepted when the 6 bit hash function of the destination address points to a bit that is set in the hash register.
no_broadcast (BCASTDI)	5	rw	0x0	No broadcast - when set to logic one, frames addressed to the broadcast address of all ones will not be accepted.
copy_all (COPYALLEN)	4	rw	0x0	Copy all frames - when set to logic one, all valid frames will be accepted.

Field Name	Bits	Type	Reset Value	Description
reserved	3	rw	0x0	Reserved. Do not modify.
disc_non_vlan (NVLANDISC)	2	rw	0x0	Discard non-VLAN frames - when set only VLAN tagged frames will be passed to the address matching logic.
full_duplex (FDEN)	1	rw	0x0	Full duplex - if set to logic one, the transmit block ignores the state of collision and carrier sense and allows receive while transmitting. Also controls the half-duplex pin.
speed (100)	0	rw	0x0	Speed - set to logic one to indicate 100Mbps operation, logic zero for 10Mbps. The value of this pin is reflected on the speed_mode[0] output pin.

Register (GEM) net_status

Name	net_status
Software Name	XEMACPS_NWSR
Relative Address	0x00000008
Absolute Address	gem0: 0xE000B008 gem1: 0xE000C008
Width	32 bits
Access Type	ro
Reset Value	x
Description	Network Status

Register net_status Details

The network status register returns status information with respect to the PHY management interface.

Field Name	Bits	Type	Reset Value	Description
reserved	31:7	ro	0x0	Reserved, read as zero, ignored on write.
pfc_pri_pause_neg	6	ro	0x0	Set when PFC Priority Based Pause has been negotiated.
pcs_autoneg_pause_tx_res	5	ro	0x0	PCS auto-negotiation pause transmit resolution.
pcs_autoneg_pause_rx_res	4	ro	0x0	PCS auto-negotiation pause receive resolution
pcs_autoneg_dup_res	3	ro	0x0	PCS auto-negotiation duplex resolution. Set to one if the resolution function determines that both devices are capable of full duplex operation. If zero half-duplex operation is possible as long as bit 0 (PCS link state) is also one.

Field Name	Bits	Type	Reset Value	Description
phy_mgmt_idle	2	ro	0x1	The PHY management logic is idle (i.e. has completed).
mdio_in_pin_status (MDIO)	1	ro	x	Returns status of the mdio_in pin
pcs_link_state	0	ro	0x0	Returns status of PCS link state. If auto-negotiation is disabled this returns the synchronization status. If auto-negotiation is enabled it is set in the LINK_OK state as long as a compatible duplex mode is resolved, it is always set in the LINK_OK state in SGMII mode.

Register ([GEM](#)) user_io

Name	user_io
Relative Address	0x0000000C
Absolute Address	gem0: 0xE000B00C gem1: 0xE000C00C
Width	32 bits
Access Type	mixed
Reset Value	x
Description	User Input/Output

Register user_io Details

The GEM design provides up to 16 inputs and 16 outputs so that the I/O can be read or set under the control of the processor interface.

If the user I/O is disabled as a configuration option, this address space is defined as reserved, and hence will be a read-only register of the value 0x0.

If enabled, the number of inputs and outputs can be configured separately. The first output will be represented in bit 0 of the user I/O register, the second output will use bit 1 and so on.

The first input will be represented in bit 16 of the user I/O register, the second input will use bit 17 and so on.

Field Name	Bits	Type	Reset Value	Description
user_in	31:16	ro	x	User programmable inputs - the upper 16 bits of this register are used to monitor the state of the user inputs. A logic one read from a bit in this range will correspond to the input being in a high state. A logic zero read from a bit in this range will correspond to the input being in a low state. Any unused bits will be read as zero. Writing to any bits in this range will have no functional effect.
user_out	15:0	rw	0x0	User programmable outputs - the lower 16 bits of this register are used to control the state of the user outputs. A logic one written to a bit in this range will cause the corresponding output to be set high. A logic zero written to a bit in this range shall cause the corresponding output to be forced low. Any unused bits will be read as logic zero. Writing to any unused bits in this range will have no functional effect.

Register ([GEM](#)) dma_cfg

Name	dma_cfg
Software Name	XEMACPS_DMACR
Relative Address	0x00000010
Absolute Address	gem0: 0xE000B010 gem1: 0xE000C010
Width	32 bits
Access Type	mixed
Reset Value	0x00020784
Description	DMA Configuration

Register dma_cfg Details

DMA Configuration

Field Name	Bits	Type	Reset Value	Description
reserved	31:25	ro	0x0	Reserved, read as zero, ignored on write.
disc_when_no_ahb	24	rw	0x0	When set, the GEM DMA will automatically discard receive packets from the receiver packet buffer memory when no AHB resource is available. When low, then received packets will remain to be stored in the SRAM based packet buffer until AHB buffer resource next becomes available.

Field Name	Bits	Type	Reset Value	Description
ahb_mem_rx_buf_size (RXBUF)	23:16	rw	0x2	<p>DMA receive buffer size in AHB system memory. The value defined by these bits determines the size of buffer to use in main AHB system memory when writing received data.</p> <p>The value is defined in multiples of 64 bytes such that a value of 0x01 corresponds to buffers of 64 bytes, 0x02 corresponds to 128 bytes etc.</p> <p>For example:</p> <p>0x02: 128 byte</p> <p>0x18: 1536 byte (1*max length frame/buffer)</p> <p>0xA0: 10240 byte (1*10k jumbo frame/buffer)</p> <p>Note that this value should never be written as zero.</p>
reserved	15:12	ro	0x0	Reserved, read as zero, ignored on write.
csum_gen_offload_en (TCPCKSUM)	11	rw	0x0	<p>Transmitter IP, TCP and UDP checksum generation offload enable. When set, the transmitter checksum generation engine is enabled, to calculate and substitute checksums for transmit frames. When clear, frame data is unaffected.</p> <p>If the GEM is not configured to use the DMA packet buffer, this bit is not implemented and will be treated as reserved, read as zero, ignored on write.</p> <p>Zynq uses packet buffer.</p>
tx_pktbuf_memsz_sel (TXSIZE)	10	rw	0x1	<p>Transmitter packet buffer memory size select - Having this bit at zero halves the amount of memory used for the transmit packet buffer. This reduces the amount of memory used by the GEM. It is important to set this bit to one if the full configured physical memory is available. The value in brackets below represents the size that would result for the default maximum configured memory size of 4 kB.</p> <p>1: Use full configured addressable space (4 kB)</p> <p>0: Do not use top address bit (2 kB)</p> <p>If the GEM is not configured to use the DMA packet buffer, this bit is not implemented and will be treated as reserved, read as zero, ignored on write. Zynq uses packet buffer.</p>

Field Name	Bits	Type	Reset Value	Description
rx_pktbuf_memsz_sel (RXSIZE)	9:8	rw	0x3	Receiver packet buffer memory size select - Having these bits at less than 11 reduces the amount of memory used for the receive packet buffer. This reduces the amount of memory used by the GEM. It is important to set these bits both to one if the full configured physical memory is available. The value in brackets below represents the size that would result for the default maximum configured memory size of 8 kBs. 00: Do not use top three address bits (1 kB) 01: Do not use top two address bits (2 kB) 10: Do not use top address bit (4 kB) 11: Use full configured addressable space (8 kB) If the controller is not configured to use the DMA packet buffer, these bits are not implemented and will be treated as reserved, read as zero, ignored on write. Zynq uses packet buffer.
ahb_endian_swp_pkt_en (ENDIAN)	7	rw	0x1	AHB endian swap mode enable for packet data accesses - When set, selects swapped endianism for AHB transfers. When clear, selects little endian mode.
ahb_endian_swp_mgm_t_en	6	rw	0x0	AHB endian swap mode enable for management descriptor accesses - When set, selects swapped endianism for AHB transfers. When clear, selects little endian mode.
reserved	5	rw	0x0	Reserved, read as zero, ignored on write
ahb_fixed_burst_len (BLENGTH)	4:0	rw	0x4	AHB fixed burst length for DMA data operations - Selects the burst length to attempt to use on the AHB when transferring frame data. Not used for DMA management operations and only used where space and data size allow. Otherwise SINGLE type AHB transfers are used. Upper bits become non-writeable if the configured DMA TX and RX FIFO sizes are smaller than required to support the selected burst size. One-hot priority encoding enforced automatically on register writes as follows, where 'x' represents don't care:- 00001: Always use SINGLE AHB bursts 0001x: Always use SINGLE AHB bursts 001xx: Attempt to use INCR4 AHB bursts (default) 01xxx: Attempt to use INCR8 AHB bursts 1xxxx: Attempt to use INCR16 AHB bursts others: reserved

Register (GEM) tx_status

Name	tx_status
Software Name	XEMACPS_TXSR
Relative Address	0x00000014
Absolute Address	gem0: 0xE000B014 gem1: 0xE000C014
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Transmit Status

Register tx_status Details

This register, when read, provides details of the status of a transmit. Once read, individual bits may be cleared by writing 1 to them. It is not possible to set a bit to 1 by writing to the register.

Field Name	Bits	Type	Reset Value	Description
reserved	31:9	ro	0x0	Reserved, read as zero, ignored on write.
hresp_not_ok (HRESPNOK)	8	rw	0x0	Hresp not OK - set when the DMA block sees hresp not OK. Cleared by writing a one to this bit.
late_collision	7	rw	0x0	Late collision occurred - only set if the condition occurs in gigabit mode, as retry is not attempted. Cleared by writing a one to this bit.
tx_under_run (URUN)	6	rw	0x0	<p>Transmit under run - this bit is set if the transmitter was forced to terminate a frame that it had already began transmitting due to further data being unavailable.</p> <p>This bit is set if a transmitter status write back has not completed when another status write back is attempted.</p> <p>When using the DMA interface configured for internal FIFO mode, this bit is also set when the transmit DMA has written the SOP data into the FIFO and either the AHB bus was not granted in time for further data, or because an AHB not OK response was returned, or because a used bit was read.</p> <p>When using the DMA interface configured for packet buffer mode, this bit will never be set.</p> <p>When using the external FIFO interface, this bit is also set when the tx_r_underflow input is asserted during a frame transfer. Cleared by writing a 1.</p>
tx_complete (TXCOMPL)	5	rw	0x0	Transmit complete - set when a frame has been transmitted. Cleared by writing a one to this bit.

Field Name	Bits	Type	Reset Value	Description
tx_corr_ahb_err (BUFEXH)	4	rw	0x0	Transmit frame corruption due to AHB error - set if an error occurs whilst midway through reading transmit frame from the AHB, including HRESP errors and buffers exhausted mid frame (if the buffers run out during transmission of a frame then transmission stops, FCS shall be bad and tx_er asserted). Also set in DMA packet buffer mode if single frame is too large for configured packet buffer memory size. Cleared by writing a one to this bit.
tx_go (TXGO)	3	ro	0x0	Transmit go - if high transmit is active. When using the exposed FIFO interface, this bit represents bit 3 of the network control register. When using the DMA interface this bit represents the tx_go variable as specified in the transmit buffer description.
retry_limit_exceeded (RXOVR)	2	rw	0x0	Retry limit exceeded - cleared by writing a one to this bit.
collision (FRAMERX)	1	rw	0x0	Collision occurred - set by the assertion of collision. Cleared by writing a one to this bit. When operating in 10/100 mode, this status indicates either a collision or a late collision. In gigabit mode, this status is not set for a late collision.
used_bit_read (USEDREAD)	0	rw	0x0	Used bit read - set when a transmit buffer descriptor is read with its used bit set. Cleared by writing a one to this bit.

Register ([GEM](#)) rx_qbar

Name	rx_qbar
Software Name	XEMACPS_RXQBASE
Relative Address	0x00000018
Absolute Address	gem0: 0xE000B018 gem1: 0xE000C018
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Receive Buffer Queue Base Address

Register rx_qbar Details

This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the 'used' bits.

In terms of AMBA AHB operation, the descriptors are read from memory using a single 32bit AHB access. When the datapath is configured at 64bit, the descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single AHB access.

For 32 bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are written to using two individual non sequential accesses.

Field Name	Bits	Type	Reset Value	Description
rx_q_baseaddr	31:2	rw	0x0	Receive buffer queue base address - written with the address of the start of the receive queue.
reserved	1:0	ro	0x0	Reserved, read as 0, ignored on write.

Register ([GEM](#)) tx_qbar

Name	tx_qbar
Software Name	XEMACPS_TXQBASE
Relative Address	0x0000001C
Absolute Address	gem0: 0xE000B01C gem1: 0xE000C01C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Transmit Buffer Queue Base Address

Register tx_qbar Details

Transmit Buffer Queue Base Address

Field Name	Bits	Type	Reset Value	Description
tx_q_base_addr	31:2	rw	0x0	Transmit buffer queue base address - written with the address of the start of the transmit queue.
reserved	1:0	ro	0x0	Reserved, read as 0, ignored on write.

Register (GEM) rx_status

Name	rx_status
Software Name	XEMACPS_RXSR
Relative Address	0x00000020
Absolute Address	gem0: 0xE000B020 gem1: 0xE000C020
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Receive Status

Register rx_status Details

When read provides details of the status of a receive. Once read, individual bits may be cleared by writing 1 to them. It is not possible to set a bit to 1 by writing to the register.

Field Name	Bits	Type	Reset Value	Description
reserved	31:4	ro	0x0	Reserved, read as 0, ignored on write.
hresp_not_ok (HRESPNOK)	3	rw	0x0	Hresp not OK - set when the DMA block sees hresp not OK. Cleared by writing a one to this bit.
rx_overnun (RXOVR)	2	rw	0x0	Receive overrun - this bit is set if either the gem_dma RX FIFO or external RX FIFO were unable to store the receive frame due to a FIFO overflow, or if the receive status, reported by the gem_rx module to the gem_dma was not taken at end of frame. This bit is also set in DMA packet buffer mode if the packet buffer overflows. For DMA operation the buffer will be recovered if an overrun occurs. This bit is cleared by writing a one to it.
frame_recd (FRAMERX)	1	rw	0x0	Frame received - one or more frames have been received and placed in memory. Cleared by writing a one to this bit.
buffer_not_avail (BUFFNA)	0	rw	0x0	Buffer not available - an attempt was made to get a new buffer and the pointer indicated that it was owned by the processor. The DMA will reread the pointer each time an end of frame is received until a valid pointer is found. This bit is set following each descriptor read attempt that fails, even if consecutive pointers are unsuccessful and software has in the mean time cleared the status flag. Cleared by writing a one to this bit.

Register (GEM) intr_status

Name	intr_status
Software Name	XEMACPS_ISR
Relative Address	0x00000024
Absolute Address	gem0: 0xE000B024 gem1: 0xE000C024
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Interrupt Status, normally read-only

Register intr_status Details

Indicates an interrupt is asserted by the controller and is enabled (unmasked).

0: not asserted

1: asserted (if any bit reads as a 1, then the ethernet_int signal will be asserted to the interrupt controller)

Field Name	Bits	Type	Reset Value	Description
reserved	31:27	ro	0x0	Reserved, read as 0, ignored on write.
tsu_sec_incr	26	ro	0x0	TSU seconds register increment - indicates the register has incremented.
pdelay_resp_tx (XEMACPS_IXR_PTPP STX)	25	ro	0x0	PTP pdelay_resp frame transmitted - indicates a PTP pdelay_resp frame has been transmitted.
pdelay_req_tx (XEMACPS_IXR_PTPP DRTX)	24	ro	0x0	PTP pdelay_req frame transmitted - indicates a PTP pdelay_req frame has been transmitted.
pdelay_resp_rx (XEMACPS_IXR_PTPS TX)	23	ro	0x0	PTP pdelay_resp frame received - indicates a PTP pdelay_resp frame has been received.
pdelay_req_rx (XEMACPS_IXR_PTP DRTX)	22	ro	0x0	PTP pdelay_req frame received - indicates a PTP pdelay_req frame has been received.
sync_tx (XEMACPS_IXR_PTPP SRX)	21	ro	0x0	PTP sync frame transmitted - indicates a PTP sync frame has been transmitted.
delay_req_tx (XEMACPS_IXR_PTPP DRRX)	20	ro	0x0	PTP delay_req frame transmitted - indicates a PTP delay_req frame has been transmitted.

Field Name	Bits	Type	Reset Value	Description
sync_rx (XEMACPS_IXR_PTPS RX)	19	ro	0x0	PTP sync frame received - indicates a PTP sync frame has been received.
delay_req_rx (XEMACPS_IXR_PTP DRRX)	18	ro	0x0	PTP delay_req frame received - indicates a PTP delay_req frame has been received.
partner_pg_rx	17	ro	0x0	PCS link partner page received - set when a new base page or next page is received from the link partner. The first time this interrupt is received, it will indicate base page received and subsequent reads will indicate next pages. The next page and base page registers should only be read when this interrupt is signaled. For next pages, the link partner next page register should be read first to avoid the register being over written. This interrupt also indicates when the host should write a new page into the next page register. If further next page exchange is only required by the link partner, this register should be written with a null message page (0x2001).
autoneg_complete	16	ro	0x0	PCS auto-negotiation complete - set once the internal PCS layer has completed auto-negotiation.
ext_intr	15	ro	0x0	External interrupt - set when a rising edge has been detected on the ext_interrupt_in input pin.
pause_tx (XEMACPS_IXR_PAU SETX)	14	ro	0x0	Pause frame transmitted - indicates a pause frame has been successfully transmitted after being initiated from the network control register or from the tx_pause control pin.
pause_zero (XEMACPS_IXR_PAU SEZERO)	13	ro	0x0	Pause time zero - set when either the pause time register at address 0x38 decrements to zero, or when a valid pause frame is received with a zero pause quantum field.
pause_nonzeroq_rx (XEMACPS_IXR_PAU SENZERO)	12	ro	0x0	Pause frame with non-zero pause quantum received - indicates a valid pause has been received that has a non-zero pause quantum field.
hresp_not_ok (XEMACPS_IXR_HRE SPNOK)	11	ro	0x0	Hresp not OK - set when the DMA block sees hresp not OK.
rx_overrun (XEMACPS_IXR_RXO VR)	10	ro	0x0	Receive overrun - set when the receive overrun status bit gets set.
link_chng	9	ro	0x0	Link change - set when the state of the link detected by the internal PCS changes state.
reserved	8	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
tx_complete (XEMACPS_IXR_TXCOMPL)	7	ro	0x0	Transmit complete - set when a frame has been transmitted.
tx_corrupt_ahb_err (XEMACPS_IXR_TXEXH)	6	ro	0x0	Transmit frame corruption due to AHB error - set if an error occurs while midway through reading transmit frame from the AHB, including HRESP errors and buffers exhausted mid frame (if the buffers run out during transmission of a frame then transmission stops, FCS shall be bad and tx_er asserted). Also set in DMA packet buffer mode if single frame is too large for configured packet buffer memory size. Cleared on a read.
retry_ex_late_collisn (XEMACPS_IXR_RETRY)	5	ro	0x0	Retry limit exceeded or late collision - transmit error. Late collision will only cause this status bit to be set in gigabit mode (as a retry is not attempted).
reserved	4	ro	0x0	Reserved. Do not modify.
tx_used_read (XEMACPS_IXR_TXUSED)	3	ro	0x0	TX used bit read - set when a transmit buffer descriptor is read with its used bit set.
rx_used_read (XEMACPS_IXR_RXUSED)	2	ro	0x0	RX used bit read - set when a receive buffer descriptor is read with its used bit set.
rx_complete (XEMACPS_IXR_FRAGMENTX)	1	ro	0x0	Receive complete - a frame has been stored in memory.
mgmt_sent (XEMACPS_IXR_MGMT)	0	ro	0x0	Management frame sent - the PHY maintenance register has completed its operation.

Register ([GEM](#)) intr_en

Name	intr_en
Software Name	XEMACPS_IER
Relative Address	0x00000028
Absolute Address	gem0: 0xE000B028 gem1: 0xE000C028
Width	32 bits
Access Type	wo
Reset Value	x

Description Interrupt Enable, write-only

Register intr_en Details

Enable interrupts by writing a 1 to one or more bits.

Write a 1 to enable (unmask) the interrupt.

Writing 0 has no affect on the mask bit.

When read, this register returns zero. To control interrupt masks and read status, use the interrupt status, enable, disable and mask registers together. At reset, all interrupts are disabled (masked).

Field Name	Bits	Type	Reset Value	Description
reserved	31:27	wo	x	Reserved
tsu_sec_incr	26	wo	x	Enable TSU seconds register increment interrupt
pdelay_resp_tx (XEMACPS_IXR_PTPP STX)	25	wo	x	Enable PTP pdelay_resp frame transmitted interrupt
pdelay_req_tx (XEMACPS_IXR_PTPP DRTX)	24	wo	x	Enable PTP pdelay_req frame transmitted interrupt
pdelay_resp_rx (XEMACPS_IXR_PTPS TX)	23	wo	x	Enable PTP pdelay_resp frame received interrupt
pdelay_req_rx (XEMACPS_IXR_PTP DRTX)	22	wo	x	Enable PTP pdelay_req frame received interrupt
sync_tx (XEMACPS_IXR_PTPP SRX)	21	wo	x	Enable PTP sync frame transmitted interrupt
delay_req_tx (XEMACPS_IXR_PTPP DRRX)	20	wo	x	Enable PTP delay_req frame transmitted interrupt
sync_rx (XEMACPS_IXR_PTPS RX)	19	wo	x	Enable PTP sync frame received interrupt
delay_req_rx (XEMACPS_IXR_PTP DRRX)	18	wo	x	Enable PTP delay_req frame received interrupt
partner_pg_rx	17	wo	x	Enable PCS link partner page received interrupt
autoneg_complete	16	wo	x	Enable PCS auto-negotiation complete interrupt
ext_intr	15	wo	x	Enable external interrupt
pause_tx (XEMACPS_IXR_PAU SETX)	14	wo	x	Enable pause frame transmitted interrupt

Field Name	Bits	Type	Reset Value	Description
pause_zero (XEMACPS_IXR_PAUSEZERO)	13	wo	x	Enable pause time zero interrupt
pause_nonzeroq (XEMACPS_IXR_PAUSENZERO)	12	wo	x	Enable pause frame with non-zero pause quantum interrupt
hresp_not_ok (XEMACPS_IXR_HRESPNOK)	11	wo	x	Enable hresp not OK interrupt
rx_overrun (XEMACPS_IXR_RXOVR)	10	wo	x	Enable receive overrun interrupt
link_chng	9	wo	x	Enable link change interrupt
reserved	8	wo	x	Not used
tx_complete (XEMACPS_IXR_TXCOMPL)	7	wo	x	Enable transmit complete interrupt
tx_corrupt_ahb_err (XEMACPS_IXR_TXEXH)	6	wo	x	Enable transmit frame corruption due to AHB error interrupt
retry_ex_late_collisn (XEMACPS_IXR_RETRY)	5	wo	x	Enable retry limit exceeded or late collision interrupt
tx_underrun (XEMACPS_IXR_URUN)	4	wo	x	Enable transmit buffer under run interrupt
tx_used_read (XEMACPS_IXR_TXUSED)	3	wo	x	Enable transmit used bit read interrupt
rx_used_read (XEMACPS_IXR_RXUSED)	2	wo	x	Enable receive used bit read interrupt
rx_complete (XEMACPS_IXR_RXMERX)	1	wo	x	Enable receive complete interrupt
mgmt_done (XEMACPS_IXR_MGMT)	0	wo	x	Enable management done interrupt

Register ([GEM](#)) intr_dis

Name	intr_dis
Software Name	XEMACPS_IDR
Relative Address	0x0000002C
Absolute Address	gem0: 0xE000B02C gem1: 0xE000C02C
Width	32 bits
Access Type	wo
Reset Value	x
Description	Interrupt Disable, write-only

Register intr_dis Details

Disable interrupts by applying a mask to one or more bits.

Write 1 to disable (mask) the interrupt.

Writing 0 has no affect on the mask bit.

When read, this register returns zero.

Field Name	Bits	Type	Reset Value	Description
reserved	31:27	wo	x	Reserved
tsu_sec_incr	26	wo	x	Disable TSU seconds register increment interrupt
pdelay_resp_tx (XEMACPS_IXR_PTPP STX)	25	wo	x	Disable PTP pdelay_resp frame transmitted interrupt
pdelay_req_tx (XEMACPS_IXR_PTPP DRTX)	24	wo	x	Disable PTP pdelay_req frame transmitted interrupt
pdelay_resp_rx (XEMACPS_IXR_PTPS TX)	23	wo	x	Disable PTP pdelay_resp frame received interrupt
pdelay_req_rx (XEMACPS_IXR_PTP DRTX)	22	wo	x	Disable PTP pdelay_req frame received interrupt
sync_tx (XEMACPS_IXR_PTPP SRX)	21	wo	x	Disable PTP sync frame transmitted interrupt
delay_req_tx (XEMACPS_IXR_PTPP DRRX)	20	wo	x	Disable PTP delay_req frame transmitted interrupt

Field Name	Bits	Type	Reset Value	Description
sync_rx (XEMACPS_IXR_PTPS RX)	19	wo	x	Disable PTP sync frame received interrupt
delay_req_rx (XEMACPS_IXR_PTP DRRX)	18	wo	x	Disable PTP delay_req frame received interrupt
partner_pg_rx	17	wo	x	Disable PCS link partner page received interrupt
autoneg_complete	16	wo	x	Disable PCS auto-negotiation complete interrupt
ext_intr	15	wo	x	Disable external interrupt
pause_tx (XEMACPS_IXR_PAU SETX)	14	wo	x	Disable pause frame transmitted interrupt
pause_zero (XEMACPS_IXR_PAU SEZERO)	13	wo	x	Disable pause time zero interrupt
pause_nonzeroq (XEMACPS_IXR_PAU SENZERO)	12	wo	x	Disable pause frame with non-zero pause quantum interrupt
hresp_not_ok (XEMACPS_IXR_HRE SPNOK)	11	wo	x	Disable hresp not OK interrupt
rx_overflow (XEMACPS_IXR_RXO VR)	10	wo	x	Disable receive overrun interrupt
link_chng	9	wo	x	Disable link change interrupt
reserved	8	wo	x	Not used
tx_complete (XEMACPS_IXR_TXC OMPL)	7	wo	x	Disable transmit complete interrupt
tx_corrupt_ahb_err (XEMACPS_IXR_TXEX H)	6	wo	x	Disable transmit frame corruption due to AHB error interrupt
retry_ex_late_collisn (XEMACPS_IXR_RETR Y)	5	wo	x	Disable retry limit exceeded or late collision interrupt
tx_underrun (XEMACPS_IXR_URU N)	4	wo	x	Disable transmit buffer under run interrupt
tx_used_read (XEMACPS_IXR_TXUS ED)	3	wo	x	Disable transmit used bit read interrupt

Field Name	Bits	Type	Reset Value	Description
rx_used_read (XEMACPS_IXR_RXUSED)	2	wo	x	Disable receive used bit read interrupt
rx_complete (XEMACPS_IXR_FRAGMENT)	1	wo	x	Disable receive complete interrupt
mgmt_done (XEMACPS_IXR_MGMT)	0	wo	x	Disable management done interrupt

Register (GEM) intr_mask

Name	intr_mask
Software Name	XEMACPS_IMR
Relative Address	0x00000030
Absolute Address	gem0: 0xE000B030 gem1: 0xE000C030
Width	32 bits
Access Type	mixed
Reset Value	x
Description	Interrupt Mask Status, normally read-only

Register intr_mask Details

Indicates the mask state of each interrupt.

0: interrupt non masked (enabled)

1: interrupt masked (disabled), reset default

All interrupts are disabled after a module reset. The interrupt masks are individually controlled using the write-only interrupt enable and disable registers.

For test purposes there is a write-only function to the interrupt mask register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register.

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	ro	0x0	Reserved
pdelay_resp_tx (XEMACPS_IXR_PTPPSTX)	25	ro,wo	x	PTP pdelay_resp frame transmitted mask.
pdelay_req_tx (XEMACPS_IXR_PTPPDRTX)	24	ro,wo	x	PTP pdelay_req frame transmitted mask.

Field Name	Bits	Type	Reset Value	Description
pdelay_resp_rx (XEMACPS_IXR_PTPS TX)	23	ro,wo	x	PTP pdelay_resp frame received mask.
pdelay_req_rx (XEMACPS_IXR_PTP DRTX)	22	ro,wo	x	PTP pdelay_req frame received mask.
sync_tx (XEMACPS_IXR_PTPP SRX)	21	ro,wo	x	PTP sync frame transmitted mask.
delay_req_tx (XEMACPS_IXR_PTPP DRRX)	20	ro,wo	x	PTP delay_req frame transmitted mask.
sync_rx (XEMACPS_IXR_PTPS RX)	19	ro,wo	x	PTP sync frame received mask.
delay_req_rx (XEMACPS_IXR_PTP DRRX)	18	ro,wo	x	PTP delay_req frame received mask.
partner_pg_rx	17	ro,wo	x	PCS link partner page mask.
autoneg_complete	16	ro,wo	0x1	PCS auto-negotiation complete interrupt mask.
ext_intr	15	ro,wo	0x1	External interrupt mask.
pause_tx (XEMACPS_IXR_PAU SETX)	14	ro,wo	0x1	Pause frame transmitted interrupt mask.
pause_zero (XEMACPS_IXR_PAU SEZERO)	13	ro,wo	0x1	Pause time zero interrupt mask.
pause_nonzeroq (XEMACPS_IXR_PAU SENZERO)	12	ro,wo	0x1	Pause frame with non-zero pause quantum interrupt mask.
hresp_not_ok (XEMACPS_IXR_HRE SPNOK)	11	ro,wo	0x1	Hresp not OK interrupt mask.
rx_overrun (XEMACPS_IXR_RXO VR)	10	ro,wo	0x1	Receive overrun interrupt mask.
link_chng	9	ro,wo	0x1	Link change interrupt mask.
reserved	8	ro,wo	0x1	Not used
tx_complete (XEMACPS_IXR_TXC OMPL)	7	ro,wo	0x1	Transmit complete interrupt mask.

Field Name	Bits	Type	Reset Value	Description
tx_corrupt_ahb_err (XEMACPS_IXR_TXEXH)	6	ro,wo	0x1	Transmit frame corruption due to AHB error interrupt
retry_ex_late_collisn (XEMACPS_IXR_RETRY)	5	ro,wo	0x1	Retry limit exceeded or late collision (gigabit mode only)
tx_underrun (XEMACPS_IXR_URUN)	4	ro,wo	0x1	Transmit buffer under run interrupt mask.
tx_used_read (XEMACPS_IXR_TXUSED)	3	ro,wo	0x1	Transmit used bit read interrupt mask.
rx_used_read (XEMACPS_IXR_RXUSED)	2	ro,wo	0x1	Receive used bit read interrupt mask.
rx_complete (XEMACPS_IXR_FRAGMENTX)	1	ro,wo	0x1	Receive complete interrupt mask.
mgmt_done (XEMACPS_IXR_MGMT)	0	ro,wo	0x1	Management done interrupt mask.

Register ([GEM](#)) phy_maint

Name	phy_maint
Software Name	XEMACPS_PHYMNTNC
Relative Address	0x00000034
Absolute Address	gem0: 0xE000B034 gem1: 0xE000C034
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	PHY Maintenance

Register phy_maint Details

The PHY maintenance register is implemented as a shift register. Writing to the register starts a shift operation, which is signaled as complete when bit-2 is set in the network status register. It takes about 2000 pclk cycles to complete, when MDC is set for pclk divide by 32 in the network configuration register. An interrupt is generated upon completion. During this time, the MSB of the register is output on the MDIO pin and the LSB updated from the MDIO pin with each MDC cycle. This causes transmission of a PHY management frame on MDIO. See Section 22.2.4.5 of the IEEE 802.3 standard. Reading during the shift

operation will return the current contents of the shift register. At the end of management operation, the bits will have shifted back to their original locations. For a read operation, the data bits will be updated with data read from the PHY. It is important to write the correct values to the register to ensure a valid PHY management frame is produced.

Field Name	Bits	Type	Reset Value	Description
reserved	31	rw	0x0	Must be written with 0.
clause_22	30	rw	0x0	Must be written to 1 for Clause 22 operation. Check you PHY's spec to see if it is clause 22 or clause 45 compliant.
operation (OP)	29:28	rw	0x0	Operation. 10 is read. 01 is write.
phy_addr (ADDR)	27:23	rw	0x0	PHY address.
reg_addr (REG)	22:18	rw	0x0	Register address - specifies the register in the PHY to access.
must_10	17:16	rw	0x0	Must be written to 10.
data (DATA)	15:0	rw	0x0	For a write operation this is written with the data to be written to the PHY. After a read operation this contains the data read from the PHY.

Register ([GEM](#)) rx_pauseq

Name	rx_pauseq
Software Name	XEMACPS_RXPAUSE
Relative Address	0x00000038
Absolute Address	gem0: 0xE000B038 gem1: 0xE000C038
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Received Pause Quantum

Register rx_pauseq Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as 0, ignored on write.
rx_pauseq	15:0	ro	0x0	Received pause quantum - stores the current value of the received pause quantum register which is decremented every 512 bit times.

Register (GEM) tx_pauseq

Name	tx_pauseq
Software Name	XEMACPS_TXPAUSE
Relative Address	0x0000003C
Absolute Address	gem0: 0xE000B03C gem1: 0xE000C03C
Width	32 bits
Access Type	rw
Reset Value	0x0000FFFF
Description	Transmit Pause Quantum

Register tx_pauseq Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	rw	0x0	Reserved, read as 0, ignored on write.
tx_pauseq	15:0	rw	0xFFFF	Transmit pause quantum - written with the pause quantum value for pause frame transmission.

Register (GEM) hash_bot

Name	hash_bot
Software Name	XEMACPS_HASHL
Relative Address	0x00000080
Absolute Address	gem0: 0xE000B080 gem1: 0xE000C080
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Hash Register Bottom [31:0]

Register hash_bot Details

The unicast hash enable and the multicast hash enable bits in the network configuration register enable the reception of hash matched frames.

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	The first 32 bits of the hash address register.

Register (GEM) hash_top

Name	hash_top
Software Name	XEMACPS_HASHH
Relative Address	0x00000084
Absolute Address	gem0: 0xE000B084 gem1: 0xE000C084
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Hash Register Top [63:32]

Register hash_top Details

The unicast hash enable and the multicast hash enable bits in the network configuration register enable the reception of hash matched frames.

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	The remaining 32 bits of the hash address register.

Register (GEM) spec_addr1_bot

Name	spec_addr1_bot
Software Name	XEMACPS_LADDR1L
Relative Address	0x00000088
Absolute Address	gem0: 0xE000B088 gem1: 0xE000C088
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Specific Address 1 Bottom [31:0]

Register spec_addr1_bot Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

Register (GEM) spec_addr1_top

Name	spec_addr1_top
Software Name	XEMACPS_LADDR1H
Relative Address	0x0000008C
Absolute Address	gem0: 0xE000B08C gem1: 0xE000C08C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Specific Address 1 Top [47:32]

Register spec_addr1_top Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as 0, ignored on write
addr_msbs	15:0	rw	0x0	Specific address 1. The most significant bits of the destination address, that is bits 47:32.

Register (GEM) spec_addr2_bot

Name	spec_addr2_bot
Software Name	XEMACPS_LADDR2L
Relative Address	0x00000090
Absolute Address	gem0: 0xE000B090 gem1: 0xE000C090
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Specific Address 2 Bottom [31:0]

Register spec_addr2_bot Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

Register (GEM) spec_addr2_top

Name	spec_addr2_top
Software Name	XEMACPS_LADDR2H
Relative Address	0x00000094
Absolute Address	gem0: 0xE000B094 gem1: 0xE000C094
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Specific Address 2 Top [47:32]

Register spec_addr2_top Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as 0, ignored on write
addr_msbs	15:0	rw	0x0	Specific address 2. The most significant bits of the destination address, that is bits 47:32.

Register (GEM) spec_addr3_bot

Name	spec_addr3_bot
Software Name	XEMACPS_LADDR3L
Relative Address	0x00000098
Absolute Address	gem0: 0xE000B098 gem1: 0xE000C098
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Specific Address 3 Bottom [31:0]

Register spec_addr3_bot Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

Register (GEM) spec_addr3_top

Name	spec_addr3_top
Software Name	XEMACPS_LADDR3H
Relative Address	0x0000009C
Absolute Address	gem0: 0xE000B09C gem1: 0xE000C09C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Specific Address 3 Top [47:32]

Register spec_addr3_top Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as 0, ignored on write
addr_msbs	15:0	rw	0x0	Specific address 3. The most significant bits of the destination address, that is bits 47:32.

Register (GEM) spec_addr4_bot

Name	spec_addr4_bot
Software Name	XEMACPS_LADDR4L
Relative Address	0x000000A0
Absolute Address	gem0: 0xE000B0A0 gem1: 0xE000C0A0
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Specific Address 4 Bottom [31:0]

Register spec_addr4_bot Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

Register (GEM) spec_addr4_top

Name	spec_addr4_top
Software Name	XEMACPS_LADDR4H
Relative Address	0x000000A4
Absolute Address	gem0: 0xE000B0A4 gem1: 0xE000C0A4
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Specific Address 4 Top [47:32]

Register spec_addr4_top Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as 0, ignored on write
addr_msbs	15:0	rw	0x0	Specific address 4. The most significant bits of the destination address, that is bits 47:32.

Register (GEM) type_id_match1

Name	type_id_match1
Software Name	XEMACPS_MATCH1
Relative Address	0x000000A8
Absolute Address	gem0: 0xE000B0A8 gem1: 0xE000C0A8
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Type ID Match 1

Register type_id_match1 Details

Field Name	Bits	Type	Reset Value	Description
copy_en	31	rw	0x0	Enable copying of type ID match 1 matched frames
reserved	30:16	ro	0x0	Reserved, read as 0, ignored on write
type_id_match1	15:0	rw	0x0	Type ID match 1. For use in comparisons with received frames type ID/length field.

Register (GEM) type_id_match2

Name	type_id_match2
Software Name	XEMACPS_MATCH2
Relative Address	0x000000AC
Absolute Address	gem0: 0xE000B0AC gem1: 0xE000C0AC
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Type ID Match 2

Register type_id_match2 Details

Field Name	Bits	Type	Reset Value	Description
copy_en	31	rw	0x0	Enable copying of type ID match 2 matched frames
reserved	30:16	ro	0x0	Reserved, read as 0, ignored on write
type_id_match2	15:0	rw	0x0	Type ID match 2. For use in comparisons with received frames type ID/length field.

Register (GEM) type_id_match3

Name	type_id_match3
Software Name	XEMACPS_MATCH3
Relative Address	0x000000B0
Absolute Address	gem0: 0xE000B0B0 gem1: 0xE000C0B0
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Type ID Match 3

Register type_id_match3 Details

Field Name	Bits	Type	Reset Value	Description
copy_en	31	rw	0x0	Enable copying of type ID match 3 matched frames

Field Name	Bits	Type	Reset Value	Description
reserved	30:16	ro	0x0	Reserved, read as 0, ignored on write
type_id_match3	15:0	rw	0x0	Type ID match 3. For use in comparisons with received frames type ID/length field.

Register ([GEM](#)) type_id_match4

Name	type_id_match4
Software Name	XEMACPS_MATCH4
Relative Address	0x000000B4
Absolute Address	gem0: 0xE000B0B4 gem1: 0xE000C0B4
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Type ID Match 4

Register type_id_match4 Details

Field Name	Bits	Type	Reset Value	Description
copy_en	31	rw	0x0	Enable copying of type ID match 4 matched frames
reserved	30:16	ro	0x0	Reserved, read as 0, ignored on write
type_id_match4	15:0	rw	0x0	Type ID match 4. For use in comparisons with received frames type ID/length field.

Register ([GEM](#)) wake_on_lan

Name	wake_on_lan
Relative Address	0x000000B8
Absolute Address	gem0: 0xE000B0B8 gem1: 0xE000C0B8
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Wake on LAN Register

Register wake_on_lan Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:20	ro	0x0	Reserved - read 0, ignored when written
multi_hash_en	19	rw	0x0	Wake on LAN multicast hash event enable. When set multicast hash events will cause the wol output to be asserted.
spec_addr_reg1_en	18	rw	0x0	Wake on LAN specific address register 1 event enable. When set specific address 1 events will cause the wol output to be asserted.
arp_req_en	17	rw	0x0	Wake on LAN ARP request event enable. When set ARP request events will cause the wol output to be asserted.
magic_pkt_en	16	rw	0x0	Wake on LAN magic packet event enable. When set magic packet events will cause the wol output to be asserted.
arp_req_ip_addr	15:0	rw	0x0	Wake on LAN ARP request IP address. Written to define the least significant 16 bits of the target IP address that is matched to generate a Wake on LAN event. A value of zero will not generate an event, even if this is matched by the received frame.

Register ([GEM](#)) ipg_stretch

Name	ipg_stretch
Software Name	XEMACPS_STRETCH
Relative Address	0x000000BC
Absolute Address	gem0: 0xE000B0BC gem1: 0xE000C0BC
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	IPG stretch register

Register ipg_stretch Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as 0, ignored on write.
ipg_stretch	15:0	rw	0x0	Bits 7:0 are multiplied with the previously transmitted frame length (including preamble) bits 15:8 +1 divide the frame length. If the resulting number is greater than 96 and bit 28 is set in the network configuration register then the resulting number is used for the transmit inter-packet-gap. 1 is added to bits 15:8 to prevent a divide by zero.

Register ([GEM](#)) stacked_vlan

Name	stacked_vlan
Relative Address	0x000000C0
Absolute Address	gem0: 0xE000B0C0 gem1: 0xE000C0C0
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Stacked VLAN Register

Register stacked_vlan Details

Field Name	Bits	Type	Reset Value	Description
stacked_vlan_en	31	rw	0x0	Enable Stacked VLAN processing mode
reserved	30:16	ro	0x0	Reserved, read as 0, ignored on write.
user_def_vlan_type	15:0	rw	0x0	User defined VLAN_TYPE field. When Stacked VLAN is enabled, the first VLAN tag in a received frame will only be accepted if the VLAN type field is equal to this user defined VLAN_TYPE OR equal to the standard VLAN type (0x8100). Note that the second VLAN tag of a Stacked VLAN packet will only be matched correctly if its VLAN_TYPE field equals 0x8100.

Register ([GEM](#)) tx_pfc_pause

Name	tx_pfc_pause
Relative Address	0x000000C4

Absolute Address	gem0: 0xE000B0C4 gem1: 0xE000C0C4
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Transmit PFC Pause Register

Register tx_pfc_pause Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as 0, ignored on write.
pauseq_sel	15:8	rw	0x0	If bit 17 of the network control register is written with a one then for each entry equal to zero in the Transmit PFC Pause Register[15:8], the PFC pause frame's pause quantum field associated with that entry will be taken from the transmit pause quantum register. For each entry equal to one in the Transmit PFC Pause Register [15:8], the pause quantum associated with that entry will be zero.
pri_en_vec_val	7:0	rw	0x0	If bit 17 of the network control register is written with a one then the priority enable vector of the PFC priority based pause frame will be set equal to the value stored in this register [7:0].

Register ([GEM](#)) spec_addr1_mask_bot

Name	spec_addr1_mask_bot
Relative Address	0x000000C8
Absolute Address	gem0: 0xE000B0C8 gem1: 0xE000C0C8
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Specific Address Mask 1 Bottom [31:0]

Register spec_addr1_mask_bot Details

Field Name	Bits	Type	Reset Value	Description
mask_bits_bot	31:0	rw	0x0	Setting a bit to one masks the corresponding bit in the specific address 1 register

Register ([GEM](#)) spec_addr1_mask_top

Name	spec_addr1_mask_top
Relative Address	0x000000CC
Absolute Address	gem0: 0xE000B0CC gem1: 0xE000C0CC
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Specific Address Mask 1 Top [47:32]

Register spec_addr1_mask_top Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as 0, ignored on write.
mask_bits_top	15:0	rw	0x0	Setting a bit to one masks the corresponding bit in the specific address 1 register

Register ([GEM](#)) module_id

Name	module_id
Relative Address	0x000000FC
Absolute Address	gem0: 0xE000B0FC gem1: 0xE000C0FC
Width	32 bits
Access Type	ro
Reset Value	0x00020118
Description	Module ID

Register module_id Details

This register indicates a Cadence module identification number and module revision. The value of this register is read only as defined by `gem_revision_reg_value`. With GEM p23, it is 0x00020118.

Field Name	Bits	Type	Reset Value	Description
module_id	31:16	ro	0x2	Module identification number - for the GEM, this value is fixed at 0x0002.
module_rev	15:0	ro	0x118	Module revision - fixed byte value specific to the revision of the design which is incremented after each release of the IP. Corresponds to Zynq having GEM p23.

Register ([GEM](#)) octets_tx_bot

Name	octets_tx_bot
Software Name	XEMACPS_OCTTXL
Relative Address	0x00000100
Absolute Address	gem0: 0xE000B100 gem1: 0xE000C100
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Octets transmitted [31:0] (in frames without error)

Register octets_tx_bot Details

Bits 31:0 should be read prior to bits 47:32 to ensure reliable operation. In statistics register block. Is reset to zero on a read and stick at all ones when it counts to its maximum value. It should be read frequently enough to prevent loss of data.

For test purposes, it may be written by setting bit 7 (Write Enable) in the network control register. Setting bit 6 (increment statistics) in the network control register causes all the statistics registers to increment by one, again for test purposes.

Once a statistics register has been read, it is automatically cleared.

Field Name	Bits	Type	Reset Value	Description
octets_tx_bot	31:0	ro	0x0	Transmitted octets in frame without errors [31:0]. The number of octets transmitted in valid frames of any type. This counter is 48-bits, and is read through two registers. This count does not include octets from automatically generated pause frames.

Register ([GEM](#)) octets_tx_top

Name	octets_tx_top
Software Name	XEMACPS_OCTTXH
Relative Address	0x00000104
Absolute Address	gem0: 0xE000B104 gem1: 0xE000C104
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Octets transmitted [47:32] (in frames without error)

Register octets_tx_top Details

Bits 31:0 should be read prior to bits 47:32 to ensure reliable operation. In statistics register block. Is reset to zero on a read and stick at all ones when it counts to its maximum value. It should be read frequently enough to prevent loss of data.

For test purposes, it may be written by setting bit 7 (Write Enable) in the network control register. Setting bit 6 (increment statistics) in the network control register causes all the statistics registers to increment by one, again for test purposes.

Once a statistics register has been read, it is automatically cleared.

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as 0, ignored on write.
octets_tx_top	15:0	ro	0x0	Transmitted octets in frame without errors [47:32]. The number of octets transmitted in valid frames of any type. This counter is 48-bits, and is read through two registers. This count does not include octets from automatically generated pause frames.

Register ([GEM](#)) frames_tx

Name	frames_tx
Software Name	XEMACPS_TXCNT
Relative Address	0x00000108
Absolute Address	gem0: 0xE000B108 gem1: 0xE000C108
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Frames Transmitted, normally read-only

Register frames_tx Details

Statistical counter for Frames transmitted without an error and exclude pause frames.

NOTES for ALL Statistical registers for Frames Transferred:

The a statistical counter is read by software, it is cleared to zero by the hardware. When a counter reaches its maximum value, it stops counting and is read with all 1s. The statistical counters must be read frequently enough if data loss is to be prevented.

For test purposes, all of the statistical counters may be written to (not just read) by setting bit 7 (wren_stat_regs) in the network control register. Also for test purposes, all of the statistical counters can be incremented (by one) by writing a 1 to bit 6 (incr_stat_regs) of the network control register.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	Frames transmitted without error. A 32 bit register counting the number of frames successfully transmitted, i.e., no under run and not too many retries. Excludes pause frames.

Register ([GEM](#)) broadcast_frames_tx

Name	broadcast_frames_tx
Software Name	XEMACPS_TXBCCNT
Relative Address	0x0000010C
Absolute Address	gem0: 0xE000B10C gem1: 0xE000C10C
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Broadcast frames Tx, normally read-only

Register broadcast_frames_tx Details

Statistical counter for Broadcast Frames transmitted without an error and exclude pause frames.

Refer to the FRAMES_TX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	Broadcast frames transmitted without error. A 32 bit register counting the number of broadcast frames successfully transmitted without error, i.e., no under run and not too many retries. Excludes pause frames.

Register ([GEM](#)) multi_frames_tx

Name	multi_frames_tx
Software Name	XEMACPS_TXMCCNT
Relative Address	0x00000110
Absolute Address	gem0: 0xE000B110 gem1: 0xE000C110
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Multicast frames Tx, normally read-only

Register multi_frames_tx Details

Statistical counter for Multicast Frames transmitted without an error and exclude pause frames.

Refer to the FRAMES_TX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	Multicast frames transmitted without error. A 32 bit register counting the number of multicast frames successfully transmitted without error, i.e., no under run and not too many retries. Excludes pause frames.

Register ([GEM](#)) pause_frames_tx

Name	pause_frames_tx
Software Name	XEMACPS_TXPAUSECNT
Relative Address	0x00000114
Absolute Address	gem0: 0xE000B114 gem1: 0xE000C114
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Pause frames Tx, normally read-only

Register pause_frames_tx Details

Statistical counter for Pause Frames transmitted without an error and not sent through the FIFO interface.

Refer to the FRAMES_TX register for additional information.

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as 0, ignored on write.
pause_frames_tx	15:0	ro	0x0	Transmitted pause frames - a 16 bit register counting the number of pause frames transmitted. Only pause frames triggered by the register interface or through the external pause pins are counted as pause frames. Pause frames received through the external FIFO interface are counted in the frames transmitted counter.

Register ([GEM](#)) frames_64b_tx

Name	frames_64b_tx
Software Name	XEMACPS_TX64CNT

Relative Address	0x00000118
Absolute Address	gem0: 0xE000B118 gem1: 0xE000C118
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Frames Tx, 64-byte length

Register frames_64b_tx Details

Statistical counter of frames of 64 bytes that are transmitted without error. Does not include pause frames.

Refer to the FRAMES_TX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	64 byte frames transmitted without error. A 32 bit register counting the number of 64 byte frames successfully transmitted without error, i.e., no under run and not too many retries. Excludes pause frames.

Register ([GEM](#)) frames_65to127b_tx

Name	frames_65to127b_tx
Software Name	XEMACPS_TX65CNT
Relative Address	0x0000011C
Absolute Address	gem0: 0xE000B11C gem1: 0xE000C11C
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Frames Tx, 65 to 127-byte length

Register frames_65to127b_tx Details

Statistical counter of frames of 65 to 127 bytes that are transmitted without error. Does not include pause frames.

Refer to the FRAMES_TX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	65 to127 byte frames transmitted without error. A 32 bit register counting the number of 65 to127 byte frames successfully transmitted without error, i.e., no under run and not too many retries.

Register ([GEM](#)) frames_128to255b_tx

Name	frames_128to255b_tx
Software Name	XEMACPS_TX128CNT
Relative Address	0x00000120
Absolute Address	gem0: 0xE000B120 gem1: 0xE000C120
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Frames Tx, 128 to 255-byte length

Register frames_128to255b_tx Details

Statistical counter of frames of 128 to 255 bytes that are transmitted without error. Does not include pause frames.

Refer to the FRAMES_TX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	128 to 255 byte frames transmitted without error. A 32 bit register counting the number of 128 to 255 byte frames successfully transmitted without error, i.e., no under run and not too many retries.

Register ([GEM](#)) frames_256to511b_tx

Name	frames_256to511b_tx
Software Name	XEMACPS_TX256CNT
Relative Address	0x00000124
Absolute Address	gem0: 0xE000B124 gem1: 0xE000C124
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Frames Tx, 256 to 511-byte length

Register frames_256to511b_tx Details

Statistical counter of frames of 256 to 511 bytes that are transmitted without error. Does not include pause frames.

Refer to the FRAMES_TX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	256 to 511 byte frames transmitted without error. A 32 bit register counting the number of 256 to 511 byte frames successfully transmitted without error, i.e., no under run and not too many retries.

Register ([GEM](#)) frames_512to1023b_tx

Name	frames_512to1023b_tx
Software Name	XEMACPS_TX512CNT
Relative Address	0x00000128
Absolute Address	gem0: 0xE000B128 gem1: 0xE000C128
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Frames Tx, 512 to 1023-byte length

Register frames_512to1023b_tx Details

Statistical counter of frames of 512 to 1023 bytes that are transmitted without error. Does not include pause frames.

Refer to the FRAMES_TX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	512 to 1023 byte frames transmitted without error. A 32 bit register counting the number of 512 to 1023 byte frames successfully transmitted without error, i.e., no under run and not too many retries.

Register ([GEM](#)) frames_1024to1518b_tx

Name	frames_1024to1518b_tx
Software Name	XEMACPS_TX1024CNT
Relative Address	0x0000012C
Absolute Address	gem0: 0xE000B12C gem1: 0xE000C12C
Width	32 bits
Access Type	ro
Reset Value	0x00000000

Description Frame Tx, 1024 to 1518-byte length

Register frames_1024to1518b_tx Details

Statistical counter of frames of 1024 to 1518 bytes that are transmitted without error. Does not include pause frames.

Refer to the FRAMES_TX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	1024 to 1518 byte frames transmitted without error. A 32 bit register counting the number of 1024 to 1518 byte frames successfully transmitted without error, i.e., no under run and not too many retries.

Register ([GEM](#)) tx_under_runs

Name	tx_under_runs
Software Name	XEMACPS_TXURUNCNT
Relative Address	0x00000134
Absolute Address	gem0: 0xE000B134 gem1: 0xE000C134
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Transmit under runs

Register tx_under_runs Details

In statistics register block. Is reset to zero on a read and stick at all ones when it counts to its maximum value. It should be read frequently enough to prevent loss of data.

For test purposes, it may be written by setting bit 7 (Write Enable) in the network control register. Setting bit 6 (increment statistics) in the network control register causes all the statistics registers to increment by one, again for test purposes.

Once a statistics register has been read, it is automatically cleared.

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	ro	0x0	Reserved, read as 0, ignored on write.
tx_under_runs	9:0	ro	0x0	Transmit under runs - a 10 bit register counting the number of frames not transmitted due to a transmit under run. If this register is incremented then no other statistics register is incremented.

Register ([GEM](#)) single_collisn_frames

Name	single_collisn_frames
Software Name	XEMACPS_SNGLCOLLCNT
Relative Address	0x00000138
Absolute Address	gem0: 0xE000B138 gem1: 0xE000C138
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Single Collision Frames

Register single_collisn_frames Details

In statistics register block. Is reset to zero on a read and sticks at all ones when it counts to its maximum value. It should be read frequently enough to prevent loss of data.

For test purposes, it may be written by setting bit 7 (Write Enable) in the network control register. Setting bit 6 (increment statistics) in the network control register causes all the statistics registers to increment by one, again for test purposes.

Once a statistics register has been read, it is automatically cleared.

Field Name	Bits	Type	Reset Value	Description
reserved	31:18	ro	0x0	Reserved, read as 0, ignored on write.
single_collisn	17:0	ro	0x0	Single collision frames - an 18 bit register counting the number of frames experiencing a single collision before being successfully transmitted, i.e. no under run.

Register ([GEM](#)) multi_collisn_frames

Name	multi_collisn_frames
Software Name	XEMACPS_MULTICOLLCNT
Relative Address	0x0000013C
Absolute Address	gem0: 0xE000B13C gem1: 0xE000C13C
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Multiple Collision Frames

Register multi_collisn_frames Details

In statistics register block. Is reset to zero on a read and sticks at all ones when it counts to its maximum value. It should be read frequently enough to prevent loss of data.

For test purposes, it may be written by setting bit 7 (Write Enable) in the network control register. Setting bit 6 (increment statistics) in the network control register causes all the statistics registers to increment by one, again for test purposes.

Once a statistics register has been read, it is automatically cleared.

Field Name	Bits	Type	Reset Value	Description
reserved	31:18	ro	0x0	Reserved, read as 0, ignored on write.
multi_collisn	17:0	ro	0x0	Multiple collision frames - an 18 bit register counting the number of frames experiencing between two and fifteen collisions prior to being successfully transmitted, i.e., no under run and not too many retries.

Register ([GEM](#)) excessive_collisns

Name	excessive_collisns
Software Name	XEMACPS_EXCESSCOLLCNT
Relative Address	0x00000140
Absolute Address	gem0: 0xE000B140 gem1: 0xE000C140
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Excessive Collisions

Register excessive_collisns Details

In statistics register block. Is reset to zero on a read and sticks at all ones when it counts to its maximum value. It should be read frequently enough to prevent loss of data.

For test purposes, it may be written by setting bit 7 (Write Enable) in the network control register. Setting bit 6 (increment statistics) in the network control register causes all the statistics registers to increment by one, again for test purposes.

Once a statistics register has been read, it is automatically cleared.

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	ro	0x0	Reserved, read as 0, ignored on write.
excessive_collisns	9:0	ro	0x0	Excessive collisions - a 10 bit register counting the number of frames that failed to be transmitted because they experienced 16 collisions.

Register ([GEM](#)) late_collisns

Name	late_collisns
Software Name	XEMACPS_LATECOLLCNT
Relative Address	0x00000144
Absolute Address	gem0: 0xE000B144 gem1: 0xE000C144
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Late Collisions

Register late_collisns Details

In statistics register block. Is reset to zero on a read and sticks at all ones when it counts to its maximum value. It should be read frequently enough to prevent loss of data.

For test purposes, it may be written by setting bit 7 (Write Enable) in the network control register. Setting bit 6 (increment statistics) in the network control register causes all the statistics registers to increment by one, again for test purposes.

Once a statistics register has been read, it is automatically cleared.

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	ro	0x0	Reserved, read as 0, ignored on write.
late_collisns	9:0	ro	0x0	Late collisions - a 10 bit register counting the number of late collision occurring after the slot time (512 bits) has expired. In 10/100 mode, late collisions are counted twice i.e., both as a collision and a late collision. In gigabit mode, a late collision causes the transmission to be aborted, thus the single and multi collision registers are not updated.

Register ([GEM](#)) deferred_tx_frames

Name	deferred_tx_frames
Software Name	XEMACPS_TXDEFERCNT
Relative Address	0x00000148
Absolute Address	gem0: 0xE000B148 gem1: 0xE000C148
Width	32 bits
Access Type	ro
Reset Value	0x00000000

Description Deferred Transmission Frames

Register deferred_tx_frames Details

In statistics register block. Is reset to zero on a read and stick at all ones when it counts to its maximum value. It should be read frequently enough to prevent loss of data.

For test purposes, it may be written by setting bit 7 (Write Enable) in the network control register. Setting bit 6 (increment statistics) in the network control register causes all the statistics registers to increment by one, again for test purposes.

Once a statistics register has been read, it is automatically cleared.

Field Name	Bits	Type	Reset Value	Description
reserved	31:18	ro	0x0	Reserved, read as 0, ignored on write.
deferred_tx	17:0	ro	0x0	Deferred transmission frames - an 18 bit register counting the number of frames experiencing deferral due to carrier sense being active on their first attempt at transmission. Frames involved in any collision are not counted nor are frames that experienced a transmit under run.

Register ([GEM](#)) carrier_sense_errs

Name	carrier_sense_errs
Software Name	XEMACPS_TXCSENSECNT
Relative Address	0x0000014C
Absolute Address	gem0: 0xE000B14C gem1: 0xE000C14C
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Carrier Sense Errors.

Register carrier_sense_errs Details

In statistics register block. Is reset to zero on a read and sticks at all ones when it counts to its maximum value. It should be read frequently enough to prevent loss of data.

For test purposes, it may be written by setting bit 7 (Write Enable) in the network control register. Setting bit 6 (increment statistics) in the network control register causes all the statistics registers to increment by one, again for test purposes.

Once a statistics register has been read, it is automatically cleared.

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	ro	0x0	Reserved, read as 0, ignored on write.
carrier_sense_errs	9:0	ro	0x0	Carrier sense errors - a 10 bit register counting the number of frames transmitted where carrier sense was not seen during transmission or where carrier sense was deasserted after being asserted in a transmit frame without collision (no under run). Only incremented in half duplex mode. The only effect of a carrier sense error is to increment this register. The behavior of the other statistics registers is unaffected by the detection of a carrier sense error.

Register ([GEM](#)) octets_rx_bot

Name	octets_rx_bot
Software Name	XEMACPS_OCTRXL
Relative Address	0x00000150
Absolute Address	gem0: 0xE000B150 gem1: 0xE000C150
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Octets Received [31:0]

Register octets_rx_bot Details

Bits 31:0 should be read prior to bits 47:32 to ensure reliable operation. In statistics register block. Is reset to zero on a read and sticks at all ones when it counts to its maximum value. It should be read frequently enough to prevent loss of data.

For test purposes, it may be written by setting bit 7 (Write Enable) in the network control register. Setting bit 6 (increment statistics) in the network control register causes all the statistics registers to increment by one, again for test purposes.

Once a statistics register has been read, it is automatically cleared.

Field Name	Bits	Type	Reset Value	Description
octets_rx_bot	31:0	ro	0x0	Received octets in frame without errors [31:0]. The number of octets received in valid frames of any type. This counter is 48-bits and is read through two registers. This count does not include octets from pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Register ([GEM](#)) octets_rx_top

Name	octets_rx_top
Software Name	XEMACPS_OCTRXH
Relative Address	0x00000154
Absolute Address	gem0: 0xE000B154 gem1: 0xE000C154
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Octets Received [47:32]

Register octets_rx_top Details

Bits 31:0 should be read prior to bits 47:32 to ensure reliable operation. In statistics register block. Is reset to zero on a read and sticks at all ones when it counts to its maximum value. It should be read frequently enough to prevent loss of data.

For test purposes, it may be written by setting bit 7 (Write Enable) in the network control register. Setting bit 6 (increment statistics) in the network control register causes all the statistics registers to increment by one, again for test purposes.

Once a statistics register has been read, it is automatically cleared.

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as 0, ignored on write.
octets_rx_top	15:0	ro	0x0	Received octets in frame without errors [47:32]. The number of octets received in valid frames of any type. This counter is 48-bits and is read through two registers. This count does not include octets from pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Register ([GEM](#)) frames_rx

Name	frames_rx
Software Name	XEMACPS_RXCNT
Relative Address	0x00000158
Absolute Address	gem0: 0xE000B158 gem1: 0xE000C158
Width	32 bits
Access Type	ro
Reset Value	0x00000000

Description Frames Received, normally read-only

Register frames_rx Details

Statistical counter for Frames received without an error and exclude pause frames.

NOTES for ALL Statistical registers for Frames Transferred:

The a statistical counter is read by software, it is cleared to zero by the hardware. When a counter reaches its maximum value, it stops counting and is read with all 1s. The statistical counters must be read frequently enough if data loss is to be prevented.

For test purposes, all of the statistical counters may be written to (not just read) by setting bit 7 (wren_stat_regs) in the network control register. Also for test purposes, all of the statistical counters can be incremented (by one) by writing a 1 to bit 6 (incr_stat_regs) of the network control register.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	Frames received without error. A 32 bit register counting the number of frames successfully received. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Register ([GEM](#)) bdcast_fames_rx

Name	bdcast_fames_rx
Software Name	XEMACPS_RXBROADCNT
Relative Address	0x0000015C
Absolute Address	gem0: 0xE000B15C gem1: 0xE000C15C
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Broadcast Frames Rx

Register bdcast_fames_rx Details

Statistical counter for Broadcast Frames received without an error and exclude pause frames.

Refer to the FRAMES_RX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	Broadcast frames received without error. A 32 bit register counting the number of broadcast frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Register ([GEM](#)) multi_frames_rx

Name	multi_frames_rx
Software Name	XEMACPS_RXMULTICNT
Relative Address	0x00000160
Absolute Address	gem0: 0xE000B160 gem1: 0xE000C160
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Multicast Frames Rx

Register multi_frames_rx Details

Statistical counter for Multicast Frames received without an error and exclude pause frames.

Refer to the FRAMES_RX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	Multicast frames received without error. A 32 bit register counting the number of multicast frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Register ([GEM](#)) pause_rx

Name	pause_rx
Software Name	XEMACPS_RXPAUSECNT
Relative Address	0x00000164
Absolute Address	gem0: 0xE000B164 gem1: 0xE000C164
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Pause Frames Rx

Register pause_rx Details

Statistical counter for Pause Frames received without an error.

Refer to the FRAMES_RX register for additional information.

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as 0, ignored on write.
pause_rx	15:0	ro	0x0	Received pause frames - a 16 bit register counting the number of pause frames received without error.

Register ([GEM](#)) frames_64b_rx

Name	frames_64b_rx
Software Name	XEMACPS_RX64CNT
Relative Address	0x00000168
Absolute Address	gem0: 0xE000B168 gem1: 0xE000C168
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Frames Rx, 64-byte length

Register frames_64b_rx Details

Statistical counter for frames of 64 bytes in length that are received without an error and exclude pause frames.

Refer to the FRAMES_RX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	64 byte frames received without error. A 32 bit register counting the number of 64 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Register ([GEM](#)) frames_65to127b_rx

Name	frames_65to127b_rx
Software Name	XEMACPS_RX65CNT
Relative Address	0x0000016C
Absolute Address	gem0: 0xE000B16C gem1: 0xE000C16C
Width	32 bits
Access Type	ro

Reset Value 0x00000000

Description Frames Rx, 65 to 127-byte length

Register frames_65to127b_rx Details

Statistical counter for frames of 65 to 127 bytes in length that are received without an error and exclude pause frames.

Refer to the FRAMES_RX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	65 to 127 byte frames received without error. A 32 bit register counting the number of 65 to 127 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Register ([GEM](#)) frames_128to255b_rx

Name frames_128to255b_rx

Software Name XEMACPS_RX128CNT

Relative Address 0x00000170

Absolute Address gem0: 0xE000B170
gem1: 0xE000C170

Width 32 bits

Access Type ro

Reset Value 0x00000000

Description Frames Rx, 128 to 255-byte length

Register frames_128to255b_rx Details

Statistical counter for frames of 128 to 255 bytes in length that are received without an error and exclude pause frames.

Refer to the FRAMES_RX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	128 to 255 byte frames received without error. A 32 bit register counting the number of 128 to 255 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Register ([GEM](#)) frames_256to511b_rx

Name	frames_256to511b_rx
Software Name	XEMACPS_RX256CNT
Relative Address	0x00000174
Absolute Address	gem0: 0xE000B174 gem1: 0xE000C174
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Frames Rx, 256 to 511-byte length

Register frames_256to511b_rx Details

Statistical counter for frames of 256 to 511 bytes in length that are received without an error and exclude pause frames.

Refer to the FRAMES_RX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	256 to 511 byte frames received without error. A 32 bit register counting the number of 256 to 511 byte frames successfully transmitted without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Register ([GEM](#)) frames_512to1023b_rx

Name	frames_512to1023b_rx
Software Name	XEMACPS_RX512CNT
Relative Address	0x00000178
Absolute Address	gem0: 0xE000B178 gem1: 0xE000C178
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Frames Rx, 512 to 1023-byte length

Register frames_512to1023b_rx Details

Statistical counter for frames of 512 to 1023 bytes in length that are received without an error and exclude pause frames.

Refer to the FRAMES_RX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	512 to 1023 byte frames received without error. A 32 bit register counting the number of 512 to 1023 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Register ([GEM](#)) frames_1024to1518b_rx

Name	frames_1024to1518b_rx
Software Name	XEMACPS_RX1024CNT
Relative Address	0x0000017C
Absolute Address	gem0: 0xE000B17C gem1: 0xE000C17C
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Frames Rx, 1024 to 1518-byte length

Register frames_1024to1518b_rx Details

Statistical counter for frames of 1024 to 1518 bytes in length that are received without an error and exclude pause frames.

Refer to the FRAMES_RX register for additional information.

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	1024 to 1518 byte frames received without error. A 32 bit register counting the number of 1024 to 1518 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Register ([GEM](#)) undersz_rx

Name	undersz_rx
Software Name	XEMACPS_RXUNDRCNT
Relative Address	0x00000184
Absolute Address	gem0: 0xE000B184 gem1: 0xE000C184
Width	32 bits

Access Type ro

Reset Value 0x00000000

Description Undersize frames received

Register undersz_rx Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	ro	0x0	Reserved, read as 0, ignored on write.
undersz_rx	9:0	ro	0x0	Undersize frames received - a 10 bit register counting the number of frames received less than 64 bytes in length (10/100 mode or gigabit mode, full duplex) that do not have either a CRC error or an alignment error. In gigabit mode, half duplex, this register counts either frames not conforming to the minimum slot time of 512 bytes or frames not conforming to the minimum frame size once bursting is active.

Register ([GEM](#)) oversz_rx

Name oversz_rx

Software Name XEMACPS_RXOVRCNT

Relative Address 0x00000188

Absolute Address gem0: 0xE000B188
gem1: 0xE000C188

Width 32 bits

Access Type ro

Reset Value 0x00000000

Description Oversize frames received

Register oversz_rx Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	ro	0x0	Reserved, read as 0, ignored on write.
oversz_rx	9:0	ro	0x0	Oversize frames received - a 10 bit register counting the number of frames received exceeding 1518 bytes (1536 bytes if bit 8 is set in network configuration register) in length but do not have either a CRC error, an alignment error nor a receive symbol error.

Register (GEM) jab_rx

Name	jab_rx
Software Name	XEMACPS_RXJABCNT
Relative Address	0x0000018C
Absolute Address	gem0: 0xE000B18C gem1: 0xE000C18C
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Jabbers received

Register jab_rx Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	ro	0x0	Reserved, read as 0, ignored on write.
jab_rx	9:0	ro	0x0	Jabbers received - a 10 bit register counting the number of frames received exceeding 1518 bytes (1536 if bit 8 set in network configuration register) in length and have either a CRC error, an alignment error or a receive symbol error.

Register (GEM) fcs_errors

Name	fcs_errors
Software Name	XEMACPS_RXFCSCNT
Relative Address	0x00000190
Absolute Address	gem0: 0xE000B190 gem1: 0xE000C190
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Frame check sequence errors

Register fcs_errors Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	ro	0x0	Reserved, read as 0, ignored on write.
fcs_errors	9:0	ro	0x0	Frame check sequence errors - a 10 bit register counting frames that are an integral number of bytes, have bad CRC and are between 64 and 1518 bytes in length. This register is also incremented if a symbol error is detected and the frame is of valid length and has an integral number of bytes. This register is incremented for a frame with bad FCS, regardless of whether it is copied to memory due to ignore FCS mode being enabled in bit 26 of the network configuration register.H524

Register ([GEM](#)) length_field_errors

Name	length_field_errors
Software Name	XEMACPS_RXLENGTHCNT
Relative Address	0x00000194
Absolute Address	gem0: 0xE000B194 gem1: 0xE000C194
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Length field frame errors

Register length_field_errors Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	ro	0x0	Reserved, read as 0, ignored on write.
length_field_errors	9:0	ro	0x0	Length field frame errors - this 10-bit register counts the number of frames received that have a measured length shorter than that extracted from the length field (bytes 13 and 14). This condition is only counted if the value of the length field is less than 0x0600, the frame is not of excessive length and checking is enabled through bit 16 of the network configuration register.

Register ([GEM](#)) rx_symbol_errors

Name	rx_symbol_errors
------	------------------

Software Name	XEMACPS_RXSYMBCNT
Relative Address	0x00000198
Absolute Address	gem0: 0xE000B198 gem1: 0xE000C198
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Receive symbol errors

Register rx_symbol_errors Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	ro	0x0	Reserved, read as 0, ignored on write.
rx_symbol_errors	9:0	ro	0x0	Receive symbol errors - a 10-bit register counting the number of frames that had rx_er asserted during reception. For 10/100 mode symbol errors are counted regardless of frame length checks. For gigabit mode the frame must satisfy slot time requirements in order to count a symbol error. Additionally, in gigabit half duplex mode, carrier extension errors are also recorded. Receive symbol errors will also be counted as an FCS or alignment error if the frame is between 64 and 1518 bytes. If the frame is larger it will be recorded as a jabber error.

Register ([GEM](#)) align_errors

Name	align_errors
Software Name	XEMACPS_RXALIGNCNT
Relative Address	0x0000019C
Absolute Address	gem0: 0xE000B19C gem1: 0xE000C19C
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Alignment errors

Register align_errors Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	ro	0x0	Reserved, read as 0, ignored on write.
align_errors	9:0	ro	0x0	Alignment errors - a 10 bit register counting frames that are not an integral number of bytes long and have bad CRC when their length is truncated to an integral number of bytes and are between 64 and 1518 bytes in length. This register is also incremented if a symbol error is detected and the frame is of valid length and does not have an integral number of bytes.

Register ([GEM](#)) rx_resource_errors

Name	rx_resource_errors
Software Name	XEMACPS_RXRESERRCNT
Relative Address	0x000001A0
Absolute Address	gem0: 0xE000B1A0 gem1: 0xE000C1A0
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Receive resource errors

Register rx_resource_errors Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:18	ro	0x0	Reserved, read as 0, ignored on write.
rx_resource_errors	17:0	ro	0x0	Receive resource errors - an 18 bit register counting the number of frames that were successfully received by the MAC (correct address matched frame and adequate slot time) but could not be copied to memory because no receive buffer was available. This will be either because the AHB bus was not granted in time or because a hresp not OK was returned.

Register ([GEM](#)) rx_overflow_errors

Name	rx_overflow_errors
Software Name	XEMACPS_RXORCNT
Relative Address	0x000001A4

Absolute Address gem0: 0xE000B1A4
 gem1: 0xE000C1A4
 Width 32 bits
 Access Type ro
 Reset Value 0x00000000
 Description Receive overrun errors

Register rx_overrun_errors Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:10	ro	0x0	Reserved, read as 0, ignored on write.
rx_overrun_errors	9:0	ro	0x0	Receive overruns - a 10 bit register counting the number of frames that are address recognized but were not copied to memory due to a receive overrun.

Register ([GEM](#)) ip_hdr_csum_errors

Name ip_hdr_csum_errors
 Software Name XEMACPS_RXIPCCNT
 Relative Address 0x000001A8
 Absolute Address gem0: 0xE000B1A8
 gem1: 0xE000C1A8
 Width 32 bits
 Access Type ro
 Reset Value 0x00000000
 Description IP header checksum errors

Register ip_hdr_csum_errors Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	ro	0x0	Reserved, read as 0, ignored on write.
ip_hdr_csum_errors	7:0	ro	0x0	0 IP header checksum errors - an 8-bit register counting the number of frames discarded due to an incorrect IP header checksum, but are between 64 and 1518 bytes and do not have a CRC error, an alignment error, nor a symbol error.

Register ([GEM](#)) tcp_csum_errors

Name	tcp_csum_errors
Software Name	XEMACPS_RXTCPCCNT
Relative Address	0x000001AC
Absolute Address	gem0: 0xE000B1AC gem1: 0xE000C1AC
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	TCP checksum errors

Register tcp_csum_errors Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	ro	0x0	Reserved, read as 0, ignored on write.
tcp_csum_errors	7:0	ro	0x0	TCP checksum errors - an 8-bit register counting the number of frames discarded due to an incorrect TCP checksum, but are between 64 and 1518 bytes and do not have a CRC error, an alignment error, nor a symbol error.

Register ([GEM](#)) udp_csum_errors

Name	udp_csum_errors
Software Name	XEMACPS_RXUDPCCNT
Relative Address	0x000001B0
Absolute Address	gem0: 0xE000B1B0 gem1: 0xE000C1B0
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	UDP checksum error

Register udp_csum_errors Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	ro	0x0	Reserved, read as 0, ignored on write.
udp_csum_errors	7:0	ro	0x0	UDP checksum errors - an 8-bit register counting the number of frames discarded due to an incorrect UDP checksum, but are between 64 and 1518 bytes and do not have a CRC error, an alignment error, nor a symbol error.

Register ([GEM](#)) timer_strobe_s

Name	timer_strobe_s
Relative Address	0x000001C8
Absolute Address	gem0: 0xE000B1C8 gem1: 0xE000C1C8
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	1588 timer sync strobe seconds

Register timer_strobe_s Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	The value of the Timer Seconds register captured when gem_tsu_ms and gem_tsu_inc_ctrl are zero.

Register ([GEM](#)) timer_strobe_ns

Name	timer_strobe_ns
Relative Address	0x000001CC
Absolute Address	gem0: 0xE000B1CC gem1: 0xE000C1CC
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	1588 timer sync strobe nanoseconds

Register timer_strobe_ns Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:30	ro	0x0	Reserved, read as 0, ignored on write.
ns_reg_val	29:0	rw	0x0	The value of the Timer Nanoseconds register captured when gem_tsu_ms and gem_tsu_inc_ctrl are zero.

Register ([GEM](#)) timer_s

Name	timer_s
Software Name	XEMACPS_1588_SEC
Relative Address	0x000001D0
Absolute Address	gem0: 0xE000B1D0 gem1: 0xE000C1D0
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	1588 timer seconds

Register timer_s Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Timer count in seconds. This register is writeable. It increments by one when the 1588 nanoseconds counter counts to one second. It may also be incremented when the timer adjust register is written.

Register ([GEM](#)) timer_ns

Name	timer_ns
Software Name	XEMACPS_1588_NANOSEC
Relative Address	0x000001D4
Absolute Address	gem0: 0xE000B1D4 gem1: 0xE000C1D4
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	1588 timer nanoseconds

Register timer_ns Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:30	ro	0x0	Reserved, read as 0, ignored on write.
timer_ct_ns	29:0	rw	0x0	Timer count in nanoseconds. This register is writeable. It can also be adjusted by writes to the 1588 timer adjust register. It increments by the value of the 1588 timer increment register each clock cycle.

Register ([GEM](#)) timer_adjust

Name	timer_adjust
Software Name	XEMACPS_1588_ADJ
Relative Address	0x000001D8
Absolute Address	gem0: 0xE000B1D8 gem1: 0xE000C1D8
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	1588 timer adjust

Register timer_adjust Details

Field Name	Bits	Type	Reset Value	Description
add_subn	31	wo	0x0	Write as one to subtract from the 1588 timer. Write as zero to add to it.
reserved	30	ro	0x0	Reserved, read as 0, ignored on write.
ns_delta	29:0	wo	0x0	The number of nanoseconds to increment or decrement the 1588 timer nanoseconds register. If necessary, the 1588 seconds register will be incremented or decremented.

Register ([GEM](#)) timer_incr

Name	timer_incr
Software Name	XEMACPS_1588_INC
Relative Address	0x000001DC
Absolute Address	gem0: 0xE000B1DC gem1: 0xE000C1DC
Width	32 bits

Access Type mixed

Reset Value 0x00000000

Description 1588 timer increment

Register timer_incr Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:24	ro	0x0	Reserved, read as 0, ignored on write.
incr_b4_alt	23:16	rw	0x0	The number of increments after which the alternative increment is used.
alt_ct_ns_delta	15:8	rw	0x0	Alternative count of nanoseconds by which the 1588 timer nanoseconds register will be incremented each clock cycle.
ns_delta	7:0	rw	0x0	A count of nanoseconds by which the 1588 timer nanoseconds register will be incremented each clock cycle.

Register ([GEM](#)) ptp_tx_s

Name ptp_tx_s

Software Name XEMACPS_PTP_TXSEC

Relative Address 0x000001E0

Absolute Address gem0: 0xE000B1E0
gem1: 0xE000C1E0

Width 32 bits

Access Type ro

Reset Value 0x00000000

Description PTP event frame transmitted seconds

Register ptp_tx_s Details

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP transmit primary event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP sync or delay_req frame. An interrupt is issued when the register is updated.

Register ([GEM](#)) ptp_tx_ns

Name	ptp_tx_ns
Software Name	XEMACPS_PTP_TXNANOSEC
Relative Address	0x000001E4
Absolute Address	gem0: 0xE000B1E4 gem1: 0xE000C1E4
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	PTP event frame transmitted nanoseconds

Register ptp_tx_ns Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:30	ro	0x0	Reserved, read as 0, ignored on write.
ns_reg_val	29:0	ro	0x0	The register is updated with the value that the 1588 timer nanoseconds register held when the SFD of a PTP transmit primary event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP sync or delay_req frame. An interrupt is issued when the register is updated.

Register ([GEM](#)) ptp_rx_s

Name	ptp_rx_s
Software Name	XEMACPS_PTP_RXSEC
Relative Address	0x000001E8
Absolute Address	gem0: 0xE000B1E8 gem1: 0xE000C1E8
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	PTP event frame received seconds

Register ptp_rx_s Details

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP receive primary event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP sync or delay_req frame. An interrupt is issued when the register is updated.

Register (GEM) ptp_rx_ns

Name	ptp_rx_ns
Software Name	XEMACPS_PTP_RXNANOSEC
Relative Address	0x000001EC
Absolute Address	gem0: 0xE000B1EC gem1: 0xE000C1EC
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	PTP event frame received nanoseconds.

Register ptp_rx_ns Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:30	ro	0x0	Reserved, read as 0, ignored on write.
ns_reg_val	29:0	ro	0x0	The register is updated with the value that the 1588 timer nanoseconds register held when the SFD of a PTP receive primary event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP sync or delay_req frame. An interrupt is issued when the register is updated.

Register (GEM) ptp_peer_tx_s

Name	ptp_peer_tx_s
Software Name	XEMACPS_PTP_TXSEC
Relative Address	0x000001F0
Absolute Address	gem0: 0xE000B1F0 gem1: 0xE000C1F0
Width	32 bits

Access Type ro

Reset Value 0x00000000

Description PTP peer event frame transmitted seconds

Register ptp_peer_tx_s Details

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP transmit peer event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP pdealy_req or pdelay_resp frame. An interrupt is issued when the register is updated.

Register ([GEM](#)) ptp_peer_tx_ns

Name ptp_peer_tx_ns

Software Name XEMACPS_PTP_TXNANOSEC

Relative Address 0x000001F4

Absolute Address gem0: 0xE000B1F4
gem1: 0xE000C1F4

Width 32 bits

Access Type ro

Reset Value 0x00000000

Description PTP peer event frame transmitted nanoseconds

Register ptp_peer_tx_ns Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:30	ro	0x0	Reserved, read as 0, ignored on write.
ns_reg_val	29:0	ro	0x0	The register is updated with the value that the 1588 timer nanoseconds register held when the SFD of a PTP transmit peer event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP pdelay_req or pdelay_resp frame. An interrupt is issued when the register is updated.

Register ([GEM](#)) ptp_peer_rx_s

Name ptp_peer_rx_s

Software Name	XEMACPS_PTPP_RXSEC
Relative Address	0x000001F8
Absolute Address	gem0: 0xE000B1F8 gem1: 0xE000C1F8
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	PTP peer event frame received seconds

Register ptp_peer_rx_s Details

Field Name	Bits	Type	Reset Value	Description
	31:0	ro	0x0	The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP receive peer event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP pdelay_req or pdelay_resp frame. An interrupt is issued when the register is updated.

Register ([GEM](#)) ptp_peer_rx_ns

Name	ptp_peer_rx_ns
Software Name	XEMACPS_PTPP_RXNANOSEC
Relative Address	0x000001FC
Absolute Address	gem0: 0xE000B1FC gem1: 0xE000C1FC
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	PTP peer event frame received nanoseconds.

Register ptp_peer_rx_ns Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:30	ro	0x0	Reserved, read as 0, ignored on write.
ns_reg_val	29:0	ro	0x0	The register is updated with the value that the 1588 timer nanoseconds register held when the SFD of a PTP receive peer event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP pdelay_req or pdelay_resp frame. An interrupt is issued when the register is updated.

Register ([GEM](#)) design_cfg2

Name	design_cfg2
Relative Address	0x00000284
Absolute Address	gem0: 0xE000B284 gem1: 0xE000C284
Width	32 bits
Access Type	ro
Reset Value	x
Description	Design Configuration 2

Register design_cfg2 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:30	ro	x	Reserved. Set to zero.
gem_tx_pbuf_addr	29:26	ro	0xA	Takes the value of the `gem_tx_pbuf_addr` DEFINE. Max address bits for Tx packet buffer (10-bits for maximum 4 kB buffer) Buffer size for Tx packet buffer mode will be 4kB. This will allow one standard packet to be received while another is transferred to system memory by the DMA interface.
gem_rx_pbuf_addr	25:22	ro	0xA	Takes the value of the `gem_rx_pbuf_addr` DEFINE. Max address bits for Rx packet buffer (10-bits for maximum 4 kB buffer) Buffer size for Rx packet buffer mode will be 4kB. This will allow one standard packet to be received while another is transferred to system memory by the DMA interface.
gem_tx_pkt_buffer	21	ro	x	Takes the value of the `gem_tx_pkt_buffer` DEFINE. Defined for Zynq. Includes the transmitter packet buffer

Field Name	Bits	Type	Reset Value	Description
gem_rx_pkt_buffer	20	ro	x	Takes the value of the `gem_rx_pkt_buffer` DEFINE. Defined for Zynq. Includes the receiver packet buffer.
gem_hprot_value	19:16	ro	0x1	Takes the value of the `gem_hprot_value` DEFINE. For Zynq, set the fixed AHB HPROT value used during transfers.
gem_jumbo_max_length	15:0	ro	0x3FFF	Takes the value of the `gem_jumbo_max_length` DEFINE. Maximum length of jumbo frames accepted by receiver. This is set to the size of the smallest of the two packet buffer, minus a margin for packet headers. However, Zynq will not support jumbo frames.

Register (GEM) design_cfg3

Name	design_cfg3
Relative Address	0x00000288
Absolute Address	gem0: 0xE000B288 gem1: 0xE000C288
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Design Configuration 3

Register design_cfg3 Details

Field Name	Bits	Type	Reset Value	Description
gem_rx_base2_fifo_size	31:16	ro	0x0	Takes the value of the `gem_rx_base2_fifo_size` DEFINE. Base-2 equivalent of `gem_rx_fifo_size`
gem_rx_fifo_size	15:0	ro	0x0	Takes the value of the `gem_rx_fifo_size` DEFINE. Set the size of the small Rx FIFO for grant latency. Extended to 16 deep to allow buffering of 64 byte maximum AHB burst size in Zynq.

Register (GEM) design_cfg4

Name	design_cfg4
Relative Address	0x0000028C
Absolute Address	gem0: 0xE000B28C gem1: 0xE000C28C
Width	32 bits

Access Type ro

Reset Value 0x00000000

Description Design Configuration 4

Register design_cfg4 Details

Field Name	Bits	Type	Reset Value	Description
gem_tx_base2_fifo_size	31:16	ro	0x0	Takes the value of the `gem_tx_base2_fifo_size` DEFINE. Base-2 equivalent of `gem_tx_fifo_size`.
gem_tx_fifo_size	15:0	ro	0x0	Takes the value of the `gem_tx_fifo_size` DEFINE. Set the size of the small TX FIFO for grant latency

Register ([GEM](#)) design_cfg5

Name design_cfg5

Relative Address 0x00000290

Absolute Address gem0: 0xE000B290
gem1: 0xE000C290

Width 32 bits

Access Type ro

Reset Value x

Description Design Configuration 5

Register design_cfg5 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:29	ro	x	Reserved. Set to zero.
gem_tsu_clk	28	ro	x	Takes the value of the `gem_tsu_clk` DEFINE. Undefined in Zynq. 1588 Time Stamp Unit clock sourced from pclk rather than independent tsu_clk.
gem_rx_buffer_length_def	27:20	ro	0x2	Takes the value of the `gem_rx_buffer_length_def` DEFINE. Set the default buffer length used by Rx DMA to 128 bytes.
gem_tx_pbuf_size_def	19	ro	0x1	Takes the value of the `gem_tx_pbuf_size_def` DEFINE. Full 4 kB Tx packet buffer size - dedicated memory resource in Zynq.
gem_rx_pbuf_size_def	18:17	ro	0x3	Takes the value of the `gem_rx_pbuf_size_def` DEFINE. Full 4 kB Rx packet buffer size - dedicated memory resource in Zynq.

Field Name	Bits	Type	Reset Value	Description
gem_endian_swap_def	16:15	ro	0x2	Takes the value of the `gem_endian_swap_def` DEFINE. Set to big endian data, little endian management descriptors in Zynq.
gem_mdc_clock_div	14:12	ro	0x2	Takes the value of the `gem_mdc_clock_div` DEFINE. Set default MDC clock divisor (can still be programmed) in Zynq.
gem_dma_bus_width	11:10	ro	0x0	Takes the value of the `gem_dma_bus_width_def` DEFINE. Limit to 32-bit AHB bus in Zynq.
gem_phy_ident	9	ro	x	Takes the value of the `gem_phy_ident` DEFINE. Undefined in Zynq. Only used in PCS.
gem_tsu	8	ro	x	Takes the value of the `gem_tsu` DEFINE. Defined in Zynq. Include support for 1588 Time Stamp Unit.
gem_tx_fifo_cnt_width	7:4	ro	0x4	Takes the value of the `gem_tx_fifo_cnt_width` DEFINE. Width for `gem_tx_fifo_size`
gem_rx_fifo_cnt_width	3:0	ro	0x5	Takes the value of the `gem_rx_fifo_cnt_width` DEFINE. Width for `gem_rx_fifo_size`.

B.19 General Purpose I/O (gpio)

Module Name	General Purpose I/O (gpio)
Software Name	XGPIOPS
Base Address	0xE000A000 gpio
Description	General Purpose Input / Output
Version	1.4
Doc Version	
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
MASK_DATA_0_LSW	0x00000000	32	mixed	x	Maskable Output Data (GPIO Bank0, MIO, Lower 16bits)
MASK_DATA_0_MSW	0x00000004	32	mixed	x	Maskable Output Data (GPIO Bank0, MIO, Upper 16bits)
MASK_DATA_1_LSW	0x00000008	32	mixed	x	Maskable Output Data (GPIO Bank1, MIO, Lower 16bits)
MASK_DATA_1_MSW	0x0000000C	22	mixed	x	Maskable Output Data (GPIO Bank1, MIO, Upper 6bits)
MASK_DATA_2_LSW	0x00000010	32	mixed	0x00000000	Maskable Output Data (GPIO Bank2, EMIO, Lower 16bits)
MASK_DATA_2_MSW	0x00000014	32	mixed	0x00000000	Maskable Output Data (GPIO Bank2, EMIO, Upper 16bits)
MASK_DATA_3_LSW	0x00000018	32	mixed	0x00000000	Maskable Output Data (GPIO Bank3, EMIO, Lower 16bits)
MASK_DATA_3_MSW	0x0000001C	32	mixed	0x00000000	Maskable Output Data (GPIO Bank3, EMIO, Upper 16bits)
DATA_0	0x00000040	32	rw	x	Output Data (GPIO Bank0, MIO)
DATA_1	0x00000044	22	rw	x	Output Data (GPIO Bank1, MIO)
DATA_2	0x00000048	32	rw	0x00000000	Output Data (GPIO Bank2, EMIO)
DATA_3	0x0000004C	32	rw	0x00000000	Output Data (GPIO Bank3, EMIO)
DATA_0_RO	0x00000060	32	ro	x	Input Data (GPIO Bank0, MIO)
DATA_1_RO	0x00000064	22	ro	x	Input Data (GPIO Bank1, MIO)

Register Name	Address	Width	Type	Reset Value	Description
DATA_2_RO	0x00000068	32	ro	0x00000000	Input Data (GPIO Bank2, EMIO)
DATA_3_RO	0x0000006C	32	ro	0x00000000	Input Data (GPIO Bank3, EMIO)
DIRM_0	0x00000204	32	rw	0x00000000	Direction mode (GPIO Bank0, MIO)
OEN_0	0x00000208	32	rw	0x00000000	Output enable (GPIO Bank0, MIO)
INT_MASK_0	0x0000020C	32	ro	0x00000000	Interrupt Mask Status (GPIO Bank0, MIO)
INT_EN_0	0x00000210	32	wo	0x00000000	Interrupt Enable/Unmask (GPIO Bank0, MIO)
INT_DIS_0	0x00000214	32	wo	0x00000000	Interrupt Disable/Mask (GPIO Bank0, MIO)
INT_STAT_0	0x00000218	32	wtc	0x00000000	Interrupt Status (GPIO Bank0, MIO)
INT_TYPE_0	0x0000021C	32	rw	0xFFFFFFFF	Interrupt Type (GPIO Bank0, MIO)
INT_POLARITY_0	0x00000220	32	rw	0x00000000	Interrupt Polarity (GPIO Bank0, MIO)
INT_ANY_0	0x00000224	32	rw	0x00000000	Interrupt Any Edge Sensitive (GPIO Bank0, MIO)
DIRM_1	0x00000244	22	rw	0x00000000	Direction mode (GPIO Bank1, MIO)
OEN_1	0x00000248	22	rw	0x00000000	Output enable (GPIO Bank1, MIO)
INT_MASK_1	0x0000024C	22	ro	0x00000000	Interrupt Mask Status (GPIO Bank1, MIO)
INT_EN_1	0x00000250	22	wo	0x00000000	Interrupt Enable/Unmask (GPIO Bank1, MIO)
INT_DIS_1	0x00000254	22	wo	0x00000000	Interrupt Disable/Mask (GPIO Bank1, MIO)
INT_STAT_1	0x00000258	22	wtc	0x00000000	Interrupt Status (GPIO Bank1, MIO)
INT_TYPE_1	0x0000025C	22	rw	0x003FFFFFFF	Interrupt Type (GPIO Bank1, MIO)
INT_POLARITY_1	0x00000260	22	rw	0x00000000	Interrupt Polarity (GPIO Bank1, MIO)
INT_ANY_1	0x00000264	22	rw	0x00000000	Interrupt Any Edge Sensitive (GPIO Bank1, MIO)
DIRM_2	0x00000284	32	rw	0x00000000	Direction mode (GPIO Bank2, EMIO)

Register Name	Address	Width	Type	Reset Value	Description
OEN_2	0x00000288	32	rw	0x00000000	Output enable (GPIO Bank2, EMIO)
INT_MASK_2	0x0000028C	32	ro	0x00000000	Interrupt Mask Status (GPIO Bank2, EMIO)
INT_EN_2	0x00000290	32	wo	0x00000000	Interrupt Enable/Unmask (GPIO Bank2, EMIO)
INT_DIS_2	0x00000294	32	wo	0x00000000	Interrupt Disable/Mask (GPIO Bank2, EMIO)
INT_STAT_2	0x00000298	32	wtc	0x00000000	Interrupt Status (GPIO Bank2, EMIO)
INT_TYPE_2	0x0000029C	32	rw	0xFFFFFFFF	Interrupt Type (GPIO Bank2, EMIO)
INT_POLARITY_2	0x000002A0	32	rw	0x00000000	Interrupt Polarity (GPIO Bank2, EMIO)
INT_ANY_2	0x000002A4	32	rw	0x00000000	Interrupt Any Edge Sensitive (GPIO Bank2, EMIO)
DIRM_3	0x000002C4	32	rw	0x00000000	Direction mode (GPIO Bank3, EMIO)
OEN_3	0x000002C8	32	rw	0x00000000	Output enable (GPIO Bank3, EMIO)
INT_MASK_3	0x000002CC	32	ro	0x00000000	Interrupt Mask Status (GPIO Bank3, EMIO)
INT_EN_3	0x000002D0	32	wo	0x00000000	Interrupt Enable/Unmask (GPIO Bank3, EMIO)
INT_DIS_3	0x000002D4	32	wo	0x00000000	Interrupt Disable/Mask (GPIO Bank3, EMIO)
INT_STAT_3	0x000002D8	32	wtc	0x00000000	Interrupt Status (GPIO Bank3, EMIO)
INT_TYPE_3	0x000002DC	32	rw	0xFFFFFFFF	Interrupt Type (GPIO Bank3, EMIO)
INT_POLARITY_3	0x000002E0	32	rw	0x00000000	Interrupt Polarity (GPIO Bank3, EMIO)
INT_ANY_3	0x000002E4	32	rw	0x00000000	Interrupt Any Edge Sensitive (GPIO Bank3, EMIO)

Register ([gpio](#)) MASK_DATA_0_LSW

Name MASK_DATA_0_LSW
Software Name DATA_LSW
Relative Address 0x00000000

Absolute Address	0xE000A000
Width	32 bits
Access Type	mixed
Reset Value	x
Description	Maskable Output Data (GPIO Bank0, MIO, Lower 16bits)

Register MASK_DATA_0_LSW Details

This register enables software to change the value being output on up to 16bits at one time selectively. Only data values with a corresponding deasserted mask bit will be changed. Output data values are unchanged and hold their previous value for bits which are masked. This register avoids the need for a read-modify-write sequence for unchanged bits.

This register controls the output values for the lower 16bits of bank0, which corresponds to MIO[15:0].

Field Name	Bits	Type	Reset Value	Description
MASK_0_LSW	31:16	wo	0x0	On a write, only bits with a corresponding deasserted mask will change the output value. 0: pin value is updated 1: pin is masked Each bit controls the corresponding pin within the 16-bit half-bank. Reads return 0's.
DATA_0_LSW	15:0	rw	x	On a write, these are the data values for the corresponding GPIO output bits. Each bit controls the corresponding pin within the 16-bit half-bank. Reads return the previous value written to this register or DATA_0[15:0]. Reads do not return the value on the GPIO pin.

Register ([gpio](#)) MASK_DATA_0_MSW

Name	MASK_DATA_0_MSW
Software Name	DATA_MSW
Relative Address	0x00000004
Absolute Address	0xE000A004
Width	32 bits
Access Type	mixed
Reset Value	x
Description	Maskable Output Data (GPIO Bank0, MIO, Upper 16bits)

Register MASK_DATA_0_MSW Details

This register operates in exactly the same manner as MASK_DATA_0_LSW, except that it controls the upper 16bits of bank0, which corresponds to MIO[31:16].

Field Name	Bits	Type	Reset Value	Description
MASK_0_MSW	31:16	wo	0x0	Operation is the same as MASK_DATA_0_LSW[MASK_0_LSW]
DATA_0_MSW	15:0	rw	x	Operation is the same as MASK_DATA_0_LSW[DATA_0_LSW]

Register ([gpio](#)) MASK_DATA_1_LSW

Name	MASK_DATA_1_LSW
Relative Address	0x00000008
Absolute Address	0xE000A008
Width	32 bits
Access Type	mixed
Reset Value	x
Description	Maskable Output Data (GPIO Bank1, MIO, Lower 16bits)

Register MASK_DATA_1_LSW Details

This register operates in exactly the same manner as MASK_DATA_0_LSW, except that it controls the lower 16bits of bank1, which corresponds to MIO[47:32].

Field Name	Bits	Type	Reset Value	Description
MASK_1_LSW	31:16	wo	0x0	Operation is the same as MASK_DATA_0_LSW[MASK_0_LSW]
DATA_1_LSW	15:0	rw	x	Operation is the same as MASK_DATA_0_LSW[DATA_0_LSW]

Register ([gpio](#)) MASK_DATA_1_MSW

Name	MASK_DATA_1_MSW
Relative Address	0x0000000C
Absolute Address	0xE000A00C
Width	22 bits
Access Type	mixed
Reset Value	x
Description	Maskable Output Data (GPIO Bank1, MIO, Upper 6bits)

Register MASK_DATA_1_MSW Details

This register operates in exactly the same manner as MASK_DATA_0_LSW, except that it controls the upper 6bits of bank1, which corresponds to MIO[53:48].

NOTE: This register does not control a full 16bits because the MIO unit itself is limited to 54 pins.

Field Name	Bits	Type	Reset Value	Description
MASK_1_MSW	21:16	wo	0x0	Operation is the same as MASK_DATA_0_LSW[MASK_0_LSW]
reserved	15:6	rw	0x0	Not used, read back as zero
DATA_1_MSW	5:0	rw	x	Operation is the same as MASK_DATA_0_LSW[DATA_0_LSW]

Register ([gpio](#)) MASK_DATA_2_LSW

Name	MASK_DATA_2_LSW
Relative Address	0x00000010
Absolute Address	0xE000A010
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Maskable Output Data (GPIO Bank2, EMIO, Lower 16bits)

Register MASK_DATA_2_LSW Details

This register operates in exactly the same manner as MASK_DATA_0_LSW, except that it controls the lower 16bits of bank2, which corresponds to EMIO[15:0].

Field Name	Bits	Type	Reset Value	Description
MASK_2_LSW	31:16	wo	0x0	Operation is the same as MASK_DATA_0_LSW[MASK_0_LSW]
DATA_2_LSW	15:0	rw	0x0	Operation is the same as MASK_DATA_0_LSW[DATA_0_LSW]

Register ([gpio](#)) MASK_DATA_2_MSW

Name	MASK_DATA_2_MSW
Relative Address	0x00000014
Absolute Address	0xE000A014
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Maskable Output Data (GPIO Bank2, EMIO, Upper 16bits)

Register MASK_DATA_2_MSW Details

This register operates in exactly the same manner as MASK_DATA_0_LSW, except that it controls the upper 16bits of bank2, which corresponds to EMIO[31:16].

Field Name	Bits	Type	Reset Value	Description
MASK_2_MSW	31:16	wo	0x0	Operation is the same as MASK_DATA_0_LSW[MASK_0_LSW]
DATA_2_MSW	15:0	rw	0x0	Operation is the same as MASK_DATA_0_LSW[DATA_0_LSW]

Register ([gpio](#)) MASK_DATA_3_LSW

Name	MASK_DATA_3_LSW
Relative Address	0x00000018
Absolute Address	0xE000A018
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Maskable Output Data (GPIO Bank3, EMIO, Lower 16bits)

Register MASK_DATA_3_LSW Details

This register operates in exactly the same manner as MASK_DATA_0_LSW, except that it controls the lower 16bits of bank3, which corresponds to EMIO[47:32].

Field Name	Bits	Type	Reset Value	Description
MASK_3_LSW	31:16	wo	0x0	Operation is the same as MASK_DATA_0_LSW[MASK_0_LSW]
DATA_3_LSW	15:0	rw	0x0	Operation is the same as MASK_DATA_0_LSW[DATA_0_LSW]

Register ([gpio](#)) MASK_DATA_3_MSW

Name	MASK_DATA_3_MSW
Relative Address	0x0000001C
Absolute Address	0xE000A01C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Maskable Output Data (GPIO Bank3, EMIO, Upper 16bits)

Register MASK_DATA_3_MSW Details

This register operates in exactly the same manner as MASK_DATA_0_LSW, except that it controls the upper 16bits of bank3, which corresponds to EMIO[63:48].

Field Name	Bits	Type	Reset Value	Description
MASK_3_MSW	31:16	wo	0x0	Operation is the same as MASK_DATA_0_LSW[MASK_0_LSW]
DATA_3_MSW	15:0	rw	0x0	Operation is the same as MASK_DATA_0_LSW[DATA_0_LSW]

Register ([gpio](#)) DATA_0

Name	DATA_0
Software Name	DATA
Relative Address	0x00000040
Absolute Address	0xE000A040
Width	32 bits
Access Type	rw
Reset Value	x
Description	Output Data (GPIO Bank0, MIO)

Register DATA_0 Details

This register controls the value being output when the GPIO signal is configured as an output. All 32bits of this register are written at one time. Reading from this register returns the previous value written to either DATA or MASK_DATA_{LSW,MSW}; it does not return the value on the device pin.

This register controls the output values for bank0, which corresponds to MIO[31:0].

Field Name	Bits	Type	Reset Value	Description
DATA_0	31:0	rw	x	Output Data

Register ([gpio](#)) DATA_1

Name	DATA_1
Relative Address	0x00000044
Absolute Address	0xE000A044
Width	22 bits
Access Type	rw
Reset Value	x
Description	Output Data (GPIO Bank1, MIO)

Register DATA_1 Details

This register operates in exactly the same manner as DATA_0, except that it controls bank1, which corresponds to MIO[53:32].

Field Name	Bits	Type	Reset Value	Description
DATA_1	21:0	rw	x	Output Data

Register ([gpio](#)) DATA_2

Name	DATA_2
Relative Address	0x00000048
Absolute Address	0xE000A048
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Output Data (GPIO Bank2, EMIO)

Register DATA_2 Details

This register operates in exactly the same manner as DATA_0, except that it controls bank2, which corresponds to EMIO[31:0].

Field Name	Bits	Type	Reset Value	Description
DATA_2	31:0	rw	0x0	Output Data

Register ([gpio](#)) DATA_3

Name	DATA_3
Relative Address	0x0000004C
Absolute Address	0xE000A04C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Output Data (GPIO Bank3, EMIO)

Register DATA_3 Details

This register operates in exactly the same manner as DATA_0, except that it controls bank3, which corresponds to EMIO[63:32].

Field Name	Bits	Type	Reset Value	Description
DATA_3	31:0	rw	0x0	Output Data

Register ([gpio](#)) DATA_0_RO

Name	DATA_0_RO
Relative Address	0x00000060
Absolute Address	0xE000A060
Width	32 bits
Access Type	ro
Reset Value	x
Description	Input Data (GPIO Bank0, MIO)

Register DATA_0_RO Details

This register enables software to observe the value on the device pin. If the GPIO signal is configured as an output, then this would normally reflect the value being driven on the output. Writes to this register are ignored.

This register reflects the input values for bank0, which corresponds to MIO[31:0].

NOTE: If the MIO is not configured to enable this pin as a GPIO pin, then DATA_RO is unpredictable. In other words, software cannot observe values on non-GPIO pins through the GPIO registers.

Field Name	Bits	Type	Reset Value	Description
DATA_0_RO	31:0	ro	x	Input Data NOTE: bits[8:7] of bank0 cannot be used as inputs and will always return 0 when read. See the GPIO chapter for more information.

Register ([gpio](#)) DATA_1_RO

Name	DATA_1_RO
Relative Address	0x00000064
Absolute Address	0xE000A064
Width	22 bits
Access Type	ro
Reset Value	x
Description	Input Data (GPIO Bank1, MIO)

Register DATA_1_RO Details

This register operates in exactly the same manner as DATA_0_RO, except that it reflects bank1, which corresponds to MIO[53:32].

Field Name	Bits	Type	Reset Value	Description
DATA_1_RO	21:0	ro	x	Input Data

Register ([gpio](#)) DATA_2_RO

Name	DATA_2_RO
Relative Address	0x00000068
Absolute Address	0xE000A068
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Input Data (GPIO Bank2, EMIO)

Register DATA_2_RO Details

This register operates in exactly the same manner as DATA_0_RO, except that it reflects bank2, which corresponds to EMIO[31:0].

Field Name	Bits	Type	Reset Value	Description
DATA_2_RO	31:0	ro	0x0	Input Data

Register ([gpio](#)) DATA_3_RO

Name	DATA_3_RO
Relative Address	0x0000006C
Absolute Address	0xE000A06C
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Input Data (GPIO Bank3, EMIO)

Register DATA_3_RO Details

This register operates in exactly the same manner as DATA_0_RO, except that it reflects bank3, which corresponds to EMIO[63:32].

Field Name	Bits	Type	Reset Value	Description
DATA_3_RO	31:0	ro	0x0	Input Data

Register ([gpio](#)) DIRM_0

Name	DIRM_0
Software Name	DIRM
Relative Address	0x00000204
Absolute Address	0xE000A204

Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Direction mode (GPIO Bank0, MIO)

Register DIRM_0 Details

This register controls whether the IO pin is acting as an input or an output. Since the input logic is always enabled, this effectively enables/disables the output driver.

Each bit of the bank is independently controlled.

This register controls bank0, which corresponds to MIO[31:0].

Field Name	Bits	Type	Reset Value	Description
DIRECTION_0	31:0	rw	0x0	Direction mode 0: input 1: output Each bit configures the corresponding pin within the 32-bit bank NOTE: bits[8:7] of bank0 cannot be used as inputs. The DIRM bits can be set to 0, but reading DATA_RO does not reflect the input value. See the GPIO chapter for more information.

Register ([gpio](#)) OEN_0

Name	OEN_0
Software Name	OUTEN
Relative Address	0x00000208
Absolute Address	0xE000A208
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Output enable (GPIO Bank0, MIO)

Register OEN_0 Details

When the IO is configured as an output, this controls whether the output is enabled or not. When the output is disabled, the pin is tri-stated.

This register controls bank0, which corresponds to MIO[31:0].

Field Name	Bits	Type	Reset Value	Description
OP_ENABLE_0	31:0	rw	0x0	Output enables 0: disabled 1: enabled Each bit configures the corresponding pin within the 32-bit bank

Register ([gpio](#)) INT_MASK_0

Name	INT_MASK_0
Software Name	INTMASK
Relative Address	0x0000020C
Absolute Address	0xE000A20C
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Interrupt Mask Status (GPIO Bank0, MIO)

Register INT_MASK_0 Details

This register shows which bits are currently masked and which are un-masked/enabled. This register is read only, so masks cannot be changed here. Use INT_EN and INT_DIS to change the mask value.

This register controls bank0, which corresponds to MIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_MASK_0	31:0	ro	0x0	Interrupt mask 0: interrupt source enabled 1: interrupt source masked Each bit reports the status for the corresponding pin within the 32-bit bank

Register ([gpio](#)) INT_EN_0

Name	INT_EN_0
Software Name	INTEN
Relative Address	0x00000210
Absolute Address	0xE000A210
Width	32 bits
Access Type	wo
Reset Value	0x00000000

Description Interrupt Enable/Unmask (GPIO Bank0, MIO)

Register INT_EN_0 Details

This register is used to enable or unmask a GPIO input for use as an interrupt source. Writing a 1 to any bit of this register enables/unmasks that signal for interrupts. Reading from this register returns an unpredictable value.

This register controls bank0, which corresponds to MIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_ENABLE_0	31:0	wo	0x0	Interrupt enable 0: no change 1: clear interrupt mask Each bit configures the corresponding pin within the 32-bit bank

Register ([gpio](#)) INT_DIS_0

Name INT_DIS_0

Software Name INTDIS

Relative Address 0x00000214

Absolute Address 0xE000A214

Width 32 bits

Access Type wo

Reset Value 0x00000000

Description Interrupt Disable/Mask (GPIO Bank0, MIO)

Register INT_DIS_0 Details

This register is used to disable or mask a GPIO input for use as an interrupt source. Writing a 1 to any bit of this register disables/masks that signal for interrupts. Reading from this register returns an unpredictable value.

This register controls bank0, which corresponds to MIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_DISABLE_0	31:0	wo	0x0	Interrupt disable 0: no change 1: set interrupt mask Each bit configures the corresponding pin within the 32-bit bank

Register ([gpio](#)) INT_STAT_0

Name INT_STAT_0

Software Name	INTSTS
Relative Address	0x00000218
Absolute Address	0xE000A218
Width	32 bits
Access Type	wtc
Reset Value	0x00000000
Description	Interrupt Status (GPIO Bank0, MIO)

Register INT_STAT_0 Details

This registers shows if an interrupt event has occurred or not. Writing a 1 to a bit in this register clears the interrupt status for that bit. Writing a 0 to a bit in this register is ignored.

This register controls bank0, which corresponds to MIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_STATUS_0	31:0	wtc	0x0	Interrupt status Upon read: 0: no interrupt 1: interrupt event has occurred Upon write: 0: no action 1: clear interrupt status bit Each bit configures the corresponding pin within the 32-bit bank

Register ([gpio](#)) INT_TYPE_0

Name	INT_TYPE_0
Software Name	INTTYPE
Relative Address	0x0000021C
Absolute Address	0xE000A21C
Width	32 bits
Access Type	rw
Reset Value	0xFFFFFFFF
Description	Interrupt Type (GPIO Bank0, MIO)

Register INT_TYPE_0 Details

This register controls whether the interrupt is edge sensitive or level sensitive.

This register controls bank0, which corresponds to MIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_TYPE_0	31:0	rw	0xFFFFFFFF	Interrupt type 0: level-sensitive 1: edge-sensitive Each bit configures the corresponding pin within the 32-bit bank

Register ([gpio](#)) INT_POLARITY_0

Name	INT_POLARITY_0
Software Name	INTPOL
Relative Address	0x00000220
Absolute Address	0xE000A220
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Interrupt Polarity (GPIO Bank0, MIO)

Register INT_POLARITY_0 Details

This register controls whether the interrupt is active-low or active high (or falling-edge sensitive or rising-edge sensitive).

This register controls bank0, which corresponds to MIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_POL_0	31:0	rw	0x0	Interrupt polarity 0: active low or falling edge 1: active high or rising edge Each bit configures the corresponding pin within the 32-bit bank

Register ([gpio](#)) INT_ANY_0

Name	INT_ANY_0
Software Name	INTANY
Relative Address	0x00000224
Absolute Address	0xE000A224
Width	32 bits
Access Type	rw
Reset Value	0x00000000

Description Interrupt Any Edge Sensitive (GPIO Bank0, MIO)

Register INT_ANY_0 Details

If INT_TYPE is set to edge sensitive, then this register enables an interrupt event on both rising and falling edges. This register is ignored if INT_TYPE is set to level sensitive.

This register controls bank0, which corresponds to MIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_ON_ANY_0	31:0	rw	0x0	Interrupt edge triggering mode 0: trigger on single edge, using configured interrupt polarity 1: trigger on both edges Each bit configures the corresponding pin within the 32-bit bank

Register ([gpio](#)) DIRM_1

Name DIRM_1

Relative Address 0x00000244

Absolute Address 0xE000A244

Width 22 bits

Access Type rw

Reset Value 0x00000000

Description Direction mode (GPIO Bank1, MIO)

Register DIRM_1 Details

This register operates in exactly the same manner as DIRM_0, except that it reflects bank1, which corresponds to MIO[53:32].

Field Name	Bits	Type	Reset Value	Description
DIRECTION_1	21:0	rw	0x0	Operation is the same as DIRM_0[DIRECTION_0]

Register ([gpio](#)) OEN_1

Name OEN_1

Relative Address 0x00000248

Absolute Address 0xE000A248

Width 22 bits

Access Type rw

Reset Value 0x00000000

Description Output enable (GPIO Bank1, MIO)

Register OEN_1 Details

This register operates in exactly the same manner as OEN_0, except that it reflects bank1, which corresponds to MIO[53:32].

Field Name	Bits	Type	Reset Value	Description
OP_ENABLE_1	21:0	rw	0x0	Operation is the same as OEN_0[OP_ENABLE_0]

Register ([gpio](#)) INT_MASK_1

Name INT_MASK_1

Relative Address 0x0000024C

Absolute Address 0xE000A24C

Width 22 bits

Access Type ro

Reset Value 0x00000000

Description Interrupt Mask Status (GPIO Bank1, MIO)

Register INT_MASK_1 Details

This register operates in exactly the same manner as INT_MASK_0, except that it reflects bank1, which corresponds to MIO[53:32].

Field Name	Bits	Type	Reset Value	Description
INT_MASK_1	21:0	ro	0x0	Operation is the same as INT_MASK_0[INT_MASK_0]

Register ([gpio](#)) INT_EN_1

Name INT_EN_1

Relative Address 0x00000250

Absolute Address 0xE000A250

Width 22 bits

Access Type wo

Reset Value 0x00000000

Description Interrupt Enable/Unmask (GPIO Bank1, MIO)

Register INT_EN_1 Details

This register operates in exactly the same manner as INT_EN_0, except that it reflects bank1, which corresponds to MIO[53:32].

Field Name	Bits	Type	Reset Value	Description
INT_ENABLE_1	21:0	wo	0x0	Operation is the same as INT_EN_0[INT_ENABLE_0]

Register ([gpio](#)) INT_DIS_1

Name	INT_DIS_1
Relative Address	0x00000254
Absolute Address	0xE000A254
Width	22 bits
Access Type	wo
Reset Value	0x00000000
Description	Interrupt Disable/Mask (GPIO Bank1, MIO)

Register INT_DIS_1 Details

This register operates in exactly the same manner as INT_DIS_0, except that it reflects bank1, which corresponds to MIO[53:32].

Field Name	Bits	Type	Reset Value	Description
INT_DISABLE_1	21:0	wo	0x0	Operation is the same as INT_DIS_0[INT_DISABLE_0]

Register ([gpio](#)) INT_STAT_1

Name	INT_STAT_1
Relative Address	0x00000258
Absolute Address	0xE000A258
Width	22 bits
Access Type	wtc
Reset Value	0x00000000
Description	Interrupt Status (GPIO Bank1, MIO)

Register INT_STAT_1 Details

This register operates in exactly the same manner as INT_STAT_0, except that it reflects bank1, which corresponds to MIO[53:32].

Field Name	Bits	Type	Reset Value	Description
INT_STATUS_1	21:0	wtc	0x0	Operation is the same as INT_STAT_0[INT_STATUS_0]

Register ([gpio](#)) INT_TYPE_1

Name	INT_TYPE_1
Relative Address	0x0000025C
Absolute Address	0xE000A25C
Width	22 bits
Access Type	rw
Reset Value	0x003FFFFFFF
Description	Interrupt Type (GPIO Bank1, MIO)

Register INT_TYPE_1 Details

This register operates in exactly the same manner as INT_TYPE_0, except that it reflects bank1, which corresponds to MIO[53:32].

Field Name	Bits	Type	Reset Value	Description
INT_TYPE_1	21:0	rw	0x3FFFFFFF	Operation is the same as INT_TYPE_0[INT_TYPE_0]

Register ([gpio](#)) INT_POLARITY_1

Name	INT_POLARITY_1
Relative Address	0x00000260
Absolute Address	0xE000A260
Width	22 bits
Access Type	rw
Reset Value	0x00000000
Description	Interrupt Polarity (GPIO Bank1, MIO)

Register INT_POLARITY_1 Details

This register operates in exactly the same manner as INT_POLARITY_0, except that it reflects bank1, which corresponds to MIO[53:32].

Field Name	Bits	Type	Reset Value	Description
INT_POL_1	21:0	rw	0x0	Operation is the same as INT_POLARITY_0[INT_POL_0]

Register ([gpio](#)) INT_ANY_1

Name	INT_ANY_1
Relative Address	0x00000264

Absolute Address	0xE000A264
Width	22 bits
Access Type	rw
Reset Value	0x00000000
Description	Interrupt Any Edge Sensitive (GPIO Bank1, MIO)

Register INT_ANY_1 Details

This register operates in exactly the same manner as INT_ANY_0, except that it reflects bank1, which corresponds to MIO[53:32].

Field Name	Bits	Type	Reset Value	Description
INT_ON_ANY_1	21:0	rw	0x0	Operation is the same as INT_ANY_0[INT_ON_ANY_0]

Register ([gpio](#)) DIRM_2

Name	DIRM_2
Relative Address	0x00000284
Absolute Address	0xE000A284
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Direction mode (GPIO Bank2, EMIO)

Register DIRM_2 Details

This register operates in exactly the same manner as DIRM_0, except that it reflects bank2, which corresponds to EMIO[31:0].

Field Name	Bits	Type	Reset Value	Description
DIRECTION_2	31:0	rw	0x0	Operation is the same as DIRM_0[DIRECTION_0]

Register ([gpio](#)) OEN_2

Name	OEN_2
Relative Address	0x00000288
Absolute Address	0xE000A288
Width	32 bits
Access Type	rw
Reset Value	0x00000000

Description Output enable (GPIO Bank2, EMIO)

Register OEN_2 Details

This register operates in exactly the same manner as OEN_0, except that it reflects bank2, which corresponds to EMIO[31:0].

Field Name	Bits	Type	Reset Value	Description
OP_ENABLE_2	31:0	rw	0x0	Operation is the same as OEN_0[OP_ENABLE_0]

Register ([gpio](#)) INT_MASK_2

Name INT_MASK_2

Relative Address 0x0000028C

Absolute Address 0xE000A28C

Width 32 bits

Access Type ro

Reset Value 0x00000000

Description Interrupt Mask Status (GPIO Bank2, EMIO)

Register INT_MASK_2 Details

This register operates in exactly the same manner as INT_MASK_0, except that it reflects bank2, which corresponds to EMIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_MASK_2	31:0	ro	0x0	Operation is the same as INT_MASK_0[INT_MASK_0]

Register ([gpio](#)) INT_EN_2

Name INT_EN_2

Relative Address 0x00000290

Absolute Address 0xE000A290

Width 32 bits

Access Type wo

Reset Value 0x00000000

Description Interrupt Enable/Unmask (GPIO Bank2, EMIO)

Register INT_EN_2 Details

This register operates in exactly the same manner as INT_EN_0, except that it reflects bank2, which corresponds to EMIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_ENABLE_2	31:0	wo	0x0	Operation is the same as INT_EN_0[INT_ENABLE_0]

Register ([gpio](#)) INT_DIS_2

Name	INT_DIS_2
Relative Address	0x00000294
Absolute Address	0xE000A294
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	Interrupt Disable/Mask (GPIO Bank2, EMIO)

Register INT_DIS_2 Details

This register operates in exactly the same manner as INT_DIS_0, except that it reflects bank2, which corresponds to EMIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_DISABLE_2	31:0	wo	0x0	Operation is the same as INT_DIS_0[INT_DISABLE_0]

Register ([gpio](#)) INT_STAT_2

Name	INT_STAT_2
Relative Address	0x00000298
Absolute Address	0xE000A298
Width	32 bits
Access Type	wtc
Reset Value	0x00000000
Description	Interrupt Status (GPIO Bank2, EMIO)

Register INT_STAT_2 Details

This register operates in exactly the same manner as INT_STAT_0, except that it reflects bank2, which corresponds to EMIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_STATUS_2	31:0	wtc	0x0	Operation is the same as INT_STAT_0[INT_STATUS_0]

Register ([gpio](#)) INT_TYPE_2

Name	INT_TYPE_2
Relative Address	0x0000029C
Absolute Address	0xE000A29C
Width	32 bits
Access Type	rw
Reset Value	0xFFFFFFFF
Description	Interrupt Type (GPIO Bank2, EMIO)

Register INT_TYPE_2 Details

This register operates in exactly the same manner as INT_TYPE_0, except that it reflects bank2, which corresponds to EMIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_TYPE_2	31:0	rw	0xFFFFFFFF	Operation is the same as INT_TYPE_0[INT_TYPE_0]

Register ([gpio](#)) INT_POLARITY_2

Name	INT_POLARITY_2
Relative Address	0x000002A0
Absolute Address	0xE000A2A0
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Interrupt Polarity (GPIO Bank2, EMIO)

Register INT_POLARITY_2 Details

This register operates in exactly the same manner as INT_POLARITY_0, except that it reflects bank2, which corresponds to EMIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_POL_2	31:0	rw	0x0	Operation is the same as INT_POLARITY_0[INT_POL_0]

Register ([gpio](#)) INT_ANY_2

Name	INT_ANY_2
Relative Address	0x000002A4

Absolute Address	0xE000A2A4
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Interrupt Any Edge Sensitive (GPIO Bank2, EMIO)

Register INT_ANY_2 Details

This register operates in exactly the same manner as INT_ANY_0, except that it reflects bank2, which corresponds to EMIO[31:0].

Field Name	Bits	Type	Reset Value	Description
INT_ON_ANY_2	31:0	rw	0x0	Operation is the same as INT_ANY_0[INT_ON_ANY_0]

Register ([gpio](#)) DIRM_3

Name	DIRM_3
Relative Address	0x000002C4
Absolute Address	0xE000A2C4
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Direction mode (GPIO Bank3, EMIO)

Register DIRM_3 Details

This register operates in exactly the same manner as DIRM_0, except that it reflects bank3, which corresponds to EMIO[63:32].

Field Name	Bits	Type	Reset Value	Description
DIRECTION_3	31:0	rw	0x0	Operation is the same as DIRM_0[DIRECTION_0]

Register ([gpio](#)) OEN_3

Name	OEN_3
Relative Address	0x000002C8
Absolute Address	0xE000A2C8
Width	32 bits
Access Type	rw
Reset Value	0x00000000

Description Output enable (GPIO Bank3, EMIO)

Register OEN_3 Details

This register operates in exactly the same manner as OEN_0, except that it reflects bank3, which corresponds to EMIO[63:32].

Field Name	Bits	Type	Reset Value	Description
OP_ENABLE_3	31:0	rw	0x0	Operation is the same as OEN_0[OP_ENABLE_0]

Register ([gpio](#)) INT_MASK_3

Name INT_MASK_3

Relative Address 0x000002CC

Absolute Address 0xE000A2CC

Width 32 bits

Access Type ro

Reset Value 0x00000000

Description Interrupt Mask Status (GPIO Bank3, EMIO)

Register INT_MASK_3 Details

This register operates in exactly the same manner as INT_MASK_0, except that it reflects bank3, which corresponds to EMIO[63:32].

Field Name	Bits	Type	Reset Value	Description
INT_MASK_3	31:0	ro	0x0	Operation is the same as INT_MASK_0[INT_MASK_0]

Register ([gpio](#)) INT_EN_3

Name INT_EN_3

Relative Address 0x000002D0

Absolute Address 0xE000A2D0

Width 32 bits

Access Type wo

Reset Value 0x00000000

Description Interrupt Enable/Unmask (GPIO Bank3, EMIO)

Register INT_EN_3 Details

This register operates in exactly the same manner as INT_EN_0, except that it reflects bank3, which corresponds to EMIO[63:32].

Field Name	Bits	Type	Reset Value	Description
INT_ENABLE_3	31:0	wo	0x0	Operation is the same as INT_EN_0[INT_ENABLE_0]

Register ([gpio](#)) INT_DIS_3

Name	INT_DIS_3
Relative Address	0x000002D4
Absolute Address	0xE000A2D4
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	Interrupt Disable/Mask (GPIO Bank3, EMIO)

Register INT_DIS_3 Details

This register operates in exactly the same manner as INT_DIS_0, except that it reflects bank3, which corresponds to EMIO[63:32].

Field Name	Bits	Type	Reset Value	Description
INT_DISABLE_3	31:0	wo	0x0	Operation is the same as INT_DIS_0[INT_DISABLE_0]

Register ([gpio](#)) INT_STAT_3

Name	INT_STAT_3
Relative Address	0x000002D8
Absolute Address	0xE000A2D8
Width	32 bits
Access Type	wtc
Reset Value	0x00000000
Description	Interrupt Status (GPIO Bank3, EMIO)

Register INT_STAT_3 Details

This register operates in exactly the same manner as INT_STAT_0, except that it reflects bank3, which corresponds to EMIO[63:32].

Field Name	Bits	Type	Reset Value	Description
INT_STATUS_3	31:0	wtc	0x0	Operation is the same as INT_STAT_0[INT_STATUS_0]

Register ([gpio](#)) INT_TYPE_3

Name	INT_TYPE_3
Relative Address	0x000002DC
Absolute Address	0xE000A2DC
Width	32 bits
Access Type	rw
Reset Value	0xFFFFFFFF
Description	Interrupt Type (GPIO Bank3, EMIO)

Register INT_TYPE_3 Details

This register operates in exactly the same manner as INT_TYPE_0, except that it reflects bank3, which corresponds to EMIO[63:32].

Field Name	Bits	Type	Reset Value	Description
INT_TYPE_3	31:0	rw	0xFFFFFFFF	Operation is the same as INT_TYPE_0[INT_TYPE_0]

Register ([gpio](#)) INT_POLARITY_3

Name	INT_POLARITY_3
Relative Address	0x000002E0
Absolute Address	0xE000A2E0
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Interrupt Polarity (GPIO Bank3, EMIO)

Register INT_POLARITY_3 Details

This register operates in exactly the same manner as INT_POLARITY_0, except that it reflects bank3, which corresponds to EMIO[63:32].

Field Name	Bits	Type	Reset Value	Description
INT_POL_3	31:0	rw	0x0	Operation is the same as INT_POLARITY_0[INT_POL_0]

Register ([gpio](#)) INT_ANY_3

Name	INT_ANY_3
Relative Address	0x000002E4

Absolute Address	0xE000A2E4
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Interrupt Any Edge Sensitive (GPIO Bank3, EMIO)

Register INT_ANY_3 Details

This register operates in exactly the same manner as INT_ANY_0, except that it reflects bank3, which corresponds to EMIO[63:32].

Field Name	Bits	Type	Reset Value	Description
INT_ON_ANY_3	31:0	rw	0x0	Operation is the same as INT_ANY_0[INT_ON_ANY_0]

B.20 Interconnect QoS (qos301)

Module Name	Interconnect QoS (qos301)
Base Address	0xF8946000 gpv_qos301_cpu 0xF8947000 gpv_qos301_dmac 0xF8948000 gpv_qos301_iou
Description	The AMBA Network Interconnect Advanced Quality of Service (QoS-301) is an extension to the AMBA Network Interconnect (NIC-301) base product and provides programmable QoS facilities for attached AMBA masters. CPU-to-DDR port instance.
Version	r0p0
Doc Version	1.0
Vendor Info	ARM QoS-301

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
qos_cntl	0x0000010C	32	rw	0x00000000	The QoS control register contains the enable bits for all the regulators.
max_ot	0x00000110	32	rw	0x00000000	Maximum number of outstanding transactions
max_comb_ot	0x00000114	32	rw	0x00000000	Maximum number of combined outstanding transactions
aw_p	0x00000118	32	rw	0x00000000	AW channel peak rate
aw_b	0x0000011C	32	rw	0x00000000	AW channel burstiness allowance
aw_r	0x00000120	32	rw	0x00000000	AW channel average rate
ar_p	0x00000124	32	rw	0x00000000	AR channel peak rate
ar_b	0x00000128	32	rw	0x00000000	AR channel burstiness allowance
ar_r	0x0000012C	32	rw	0x00000000	AR channel average rate

Register ([qos301](#)) qos_cntl

Name	qos_cntl
Relative Address	0x0000010C

Absolute Address	gpv_qos301_cpu: 0xF894610C gpv_qos301_dmac: 0xF894710C gpv_qos301_iou: 0xF894810C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	The QoS control register contains the enable bits for all the regulators.

Register qos_cntl Details

By default, all of the bits are set to 0, and no regulation is enabled. Regulation only takes place when both the enable bit is set, and its corresponding regulation value is non-zero. The QoS regulators are reset whenever they are re-enabled.

Field Name	Bits	Type	Reset Value	Description
en_aware_ot	7	rw	0x0	Enable combined regulation of outstanding transactions.
en_ar_ot	6	rw	0x0	Enable regulation of outstanding read transactions.
en_aw_ot	5	rw	0x0	Enable regulation of outstanding write transactions.
reserved	4:3	rw	0x0	Reserved
en_aware_rate	2	rw	0x0	Enable combined AW / AR rate regulation.
en_ar_rate	1	rw	0x0	Enable AR rate regulation.
en_aw_rate	0	rw	0x0	Enable AW rate regulation.

Register ([qos301](#)) max_ot

Name	max_ot
Relative Address	0x00000110
Absolute Address	gpv_qos301_cpu: 0xF8946110 gpv_qos301_dmac: 0xF8947110 gpv_qos301_iou: 0xF8948110
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Maximum number of outstanding transactions

Register max_ot Details

The maximum number of outstanding transactions register enables you to program the maximum number of address requests for the AR and AW channels. The outstanding transaction limits have an integer part and a fractional part.

Field Name	Bits	Type	Reset Value	Description
ar_max_oti	29:24	rw	0x0	Integer part of max outstanding AR addresses.
ar_max_otf	23:16	rw	0x0	Fraction part of max outstanding AR addresses.
aw_max_oti	13:8	rw	0x0	Integer part of max outstanding AW addresses.
aw_max_otf	7:0	rw	0x0	Fraction part of max outstanding AW addresses.

A value of 0 for both the integer and fractional parts disables the programmable regulation so that the NIC-301 base product configuration limits apply.

A value of 0 for the fractional part programs disables the regulation of fractional outstanding transactions.

The AW and AR outstanding transaction limits are enabled when you set the corresponding en_aw_ot or en_ar_ot control bits of the QoS control register.

Register ([qos301](#)) max_comb_ot

Name	max_comb_ot
Relative Address	0x00000114
Absolute Address	gpv_qos301_cpu: 0xF8946114 gpv_qos301_dmac: 0xF8947114 gpv_qos301_iou: 0xF8948114
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Maximum number of combined outstanding transactions

Register max_comb_ot Details

The maximum combined outstanding transactions register enables you to program the maximum number of address requests for the AR and AW channels. The combined limit is applied after any individual channel limits.

Field Name	Bits	Type	Reset Value	Description
awar_max_oti	14:8	rw	0x0	Integer part of max combined outstanding AW/AR addresses.
awar_max_otf	7:0	rw	0x0	Fraction part of max combined outstanding AW/AR addresses.

A value of 0 for both the integer and fractional parts disables the programmable regulation so that the configuration limits apply.

A value of 0 for the fractional part programs disables the regulation of fractional outstanding transactions.

The regulation of the combined outstanding transaction limit also requires that you set the en_awar_ot control bit of the QoS control register.

Register ([qos301](#)) aw_p

Name	aw_p
Relative Address	0x00000118
Absolute Address	gpv_qos301_cpu: 0xF8946118 gpv_qos301_dmac: 0xF8947118 gpv_qos301_iou: 0xF8948118
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	AW channel peak rate

Register aw_p Details

Field Name	Bits	Type	Reset Value	Description
	31:24	rw	0x0	channel peak rate. 8-bit fraction of the number of transfers per cycle. A value of 0x80 (decimal 0.5) sets a rate of one transaction every 2 cycles. A value of 0x40 sets a rate of one transaction every 4 cycles, etc.

Register ([qos301](#)) aw_b

Name	aw_b
Relative Address	0x0000011C
Absolute Address	gpv_qos301_cpu: 0xF894611C gpv_qos301_dmac: 0xF894711C gpv_qos301_iou: 0xF894811C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	AW channel burstiness allowance

Register aw_b Details

Field Name	Bits	Type	Reset Value	Description
	15:0	rw	0x0	channel burstiness (integer number of transfers)

Register ([qos301](#)) aw_r

Name	aw_r
------	------

Relative Address	0x00000120
Absolute Address	gpv_qos301_cpu: 0xF8946120 gpv_qos301_dmac: 0xF8947120 gpv_qos301_iou: 0xF8948120
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	AW channel average rate

Register aw_r Details

Field Name	Bits	Type	Reset Value	Description
	31:20	rw	0x0	channel average rate. 12-bit fraction of the number of transfers per cycle. A value of 0x800 (decimal 0.5) sets a rate of one transaction every 2 cycles. A value of 0x400 sets a rate of one transaction every 4 cycles, etc.

Register ([qos301](#)) ar_p

Name	ar_p
Relative Address	0x00000124
Absolute Address	gpv_qos301_cpu: 0xF8946124 gpv_qos301_dmac: 0xF8947124 gpv_qos301_iou: 0xF8948124
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	AR channel peak rate

Register ar_p Details

Field Name	Bits	Type	Reset Value	Description
	31:24	rw	0x0	channel peak rate. 8-bit fraction of the number of transfers per cycle. A value of 0x80 (decimal 0.5) sets a rate of one transaction every 2 cycles. A value of 0x40 sets a rate of one transaction every 4 cycles, etc.

Register ([qos301](#)) ar_b

Name	ar_b
------	------

Relative Address	0x00000128
Absolute Address	gpv_qos301_cpu: 0xF8946128 gpv_qos301_dmac: 0xF8947128 gpv_qos301_iou: 0xF8948128
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	AR channel burstiness allowance

Register ar_b Details

Field Name	Bits	Type	Reset Value	Description
	15:0	rw	0x0	channel burstiness (integer number of transfers)

Register ([qos301](#)) ar_r

Name	ar_r
Relative Address	0x0000012C
Absolute Address	gpv_qos301_cpu: 0xF894612C gpv_qos301_dmac: 0xF894712C gpv_qos301_iou: 0xF894812C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	AR channel average rate

Register ar_r Details

Field Name	Bits	Type	Reset Value	Description
	31:20	rw	0x0	channel average rate. 12-bit fraction of the number of transfers per cycle. A value of 0x800 (decimal 0.5) sets a rate of one transaction every 2 cycles. A value of 0x400 sets a rate of one transaction every 4 cycles, etc.

Usage Example:

- Peak = 2 (or 1 in 128)
- Burstiness = 5
- Average = 10 (or 1 in 409)

This allows an issuing rate of 1 in 128 until the burstiness allowance of 5 outstanding transactions is reached. Then the average issuing rate of 1 in 409 takes effect until the number of outstanding transactions drops below 5.

B.21 NIC301 Address Region Control (nic301_addr_region_ctrl_registers)

Module Name	NIC301 Address Region Control (nic301_addr_region_ctrl_registers)
Software Name	XARC
Base Address	0xF8900000 gpv_trustzone
Description	AMBA Network Interconnect NIC301, Address Region Control Registers NIC301 TrustZone.
Version	1.0
Doc Version	1.0
Vendor Info	ARM

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
security_fssw_s0	0x0000001C	1	wo	0x00000000	Slave Interconnect S0 port boot secure setting
security_fssw_s1	0x00000020	1	wo	0x00000000	Slave Interconnect S1 port boot secure setting

Register ([nic301_addr_region_ctrl_registers](#)) security_fssw_s0

Name	security_fssw_s0
Relative Address	0x0000001C
Absolute Address	0xF890001C
Width	1 bits
Access Type	wo
Reset Value	0x00000000
Description	Slave Interconnect S0 port boot secure setting

Register security_fssw_s0 Details

Field Name	Bits	Type	Reset Value	Description
fssw_s0	0	wo	0x0	0 - Secure. 1 - Non-secure.

Register ([nic301_addr_region_ctrl_registers](#)) security_fssw_s1

Name	security_fssw_s1
Relative Address	0x00000020
Absolute Address	0xF8900020
Width	1 bits
Access Type	wo
Reset Value	0x00000000
Description	Slave Interconnect S1 port boot secure setting

Register security_fssw_s1 Details

Field Name	Bits	Type	Reset Value	Description
fssw_s1	0	wo	0x0	0 - Secure. 1 - Non-secure.

B.22 I2C Controller (IIC)

Module Name	I2C Controller (IIC)
Software Name	XIICPS
Base Address	0xE0004000 i2c0 0xE0005000 i2c1
Description	Inter Integrated Circuit (I2C) Instance no. 0.
Version	1.0
Doc Version	1.2
Vendor Info	Cadence IIC

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
Control_reg0	0x00000000	16	mixed	0x00000000	Control Register
Status_reg0	0x00000004	16	ro	0x00000000	Status register
I2C_address_reg0	0x00000008	16	mixed	0x00000000	IIC Address register
I2C_data_reg0	0x0000000C	16	mixed	0x00000000	IIC data register
Interrupt_status_reg0	0x00000010	16	mixed	0x00000000	IIC interrupt status register
Transfer_size_reg0	0x00000014	8	rw	0x00000000	Transfer Size Register
Slave_mon_pause_reg0	0x00000018	8	mixed	0x00000000	Slave Monitor Pause Register
Time_out_reg0	0x0000001C	8	rw	0x0000001F	Time out register
Intrpt_mask_reg0	0x00000020	16	ro	0x000002FF	Interrupt mask register
Intrpt_enable_reg0	0x00000024	16	mixed	0x00000000	Interrupt Enable Register
Intrpt_disable_reg0	0x00000028	16	mixed	0x00000000	Interrupt Disable Register

Register ([IIC](#)) Control_reg0

Name	Control_reg0
Software Name	CR
Relative Address	0x00000000
Absolute Address	i2c0: 0xE0004000 i2c1: 0xE0005000
Width	16 bits

Access Type mixed
Reset Value 0x00000000
Description Control Register

Register Control_reg0 Details

Field Name	Bits	Type	Reset Value	Description
divisor_a (DIV_A)	15:14	rw	0x0	Divisor for stage A clock divider. 0 - 3: Divides the input pclk frequency by divisor_a + 1.
divisor_b (DIV_B)	13:8	rw	0x0	Divisor for stage B clock divider. 0 -31 : Divides the output frequency from divisor_a by divisor_b + 1.
reserved	7	ro	0x0	Reserved, read as zero, ignored on write.
CLR_FIFO	6	rw	0x0	1 - initializes the FIFO to all zeros and clears the transfer size register. Automatically gets cleared on the next APB clock after being set.
SLVMON	5	rw	0x0	Slave monitor mode 1 - monitor mode. 0 - normal operation.
HOLD	4	rw	0x0	hold_bus 1 - when no more data is available for transmit or no more data can be received, hold the sclk line low until serviced by the host. 0 - allow the transfer to terminate as soon as all the data has been transmitted or received.
ACK_EN (ACKEN)	3	rw	0x0	This bit needs to be set to 1 1 - acknowledge enabled, ACK transmitted 0 - acknowledge disabled, NACK transmitted.
NEA	2	rw	0x0	Addressing mode: This bit is used in master mode only. 1 - normal (7-bit) address 0 - reserved
MS	1	rw	0x0	Overall interface mode: 1 - master 0 - slave
RW (RD_WR)	0	rw	0x0	Direction of transfer: This bit is used in master mode only. 1 - master receiver 0 - master transmitter.

Register (IIC) Status_reg0

Name	Status_reg0
Software Name	SR
Relative Address	0x00000004
Absolute Address	i2c0: 0xE0004004 i2c1: 0xE0005004
Width	16 bits
Access Type	ro
Reset Value	0x00000000
Description	Status register

Register Status_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	15:9	ro	0x0	Reserved, read as zero, ignored on write.
BA	8	ro	0x0	Bus Active 1 - ongoing transfer on the I2C bus.
RXOVF	7	ro	0x0	Receiver Overflow 1 - This bit is set whenever FIFO is full and a new byte is received. The new byte is not acknowledged and contents of the FIFO remains unchanged.
TXDV	6	ro	0x0	Transmit Data Valid - SW should not use this to determine data completion, it is the RAW value on the interface. Please use COMP in the ISR. 1 - still a byte of data to be transmitted by the interface.
RXDV	5	ro	0x0	Receiver Data Valid 1 - valid, new data to be read from the interface.
reserved	4	ro	0x0	Reserved, read as zero, ignored on write.
RXRW	3	ro	0x0	RX read_write 1 - mode of the transmission received from a master.
reserved	2:0	ro	0x0	Reserved, read as zero, ignored on write.

Register (IIC) I2C_address_reg0

Name	I2C_address_reg0
Software Name	ADDR

Relative Address	0x00000008
Absolute Address	i2c0: 0xE0004008 i2c1: 0xE0005008
Width	16 bits
Access Type	mixed
Reset Value	0x00000000
Description	IIC Address register

Register I2C_address_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	15:10	ro	0x0	Reserved, read as zero, ignored on write.
ADD (MASK)	9:0	rw	0x0	Address 0 - 1024: Normal addressing mode uses add[6:0]. Extended addressing mode uses add[9:0].

Register (IIC) I2C_data_reg0

Name	I2C_data_reg0
Software Name	DATA
Relative Address	0x0000000C
Absolute Address	i2c0: 0xE000400C i2c1: 0xE000500C
Width	16 bits
Access Type	mixed
Reset Value	0x00000000
Description	IIC data register

Register I2C_data_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	15:8	ro	0x0	
DATA (MASK)	7:0	rw	0x0	data 0 -255: When written to, the data register sets data to transmit. When read from, the data register reads the last received byte of data.

Register (IIC) Interrupt_status_reg0

Name	Interrupt_status_reg0
------	-----------------------

Software Name	ISR
Relative Address	0x00000010
Absolute Address	i2c0: 0xE0004010 i2c1: 0xE0005010
Width	16 bits
Access Type	mixed
Reset Value	0x00000000
Description	IIC interrupt status register

Register Interrupt_status_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	15:10	ro	0x0	Reserved, read as zero, ignored on write.
ARB_LOST (IXR_ARB_LOST)	9	wtc	0x0	arbitration lost 1 = master loses bus ownership during a transfer due to ongoing arbitration
reserved	8	ro	0x0	Reserved, read as zero, ignored on write.
RX_UNF (IXR_RX_UNF)	7	wtc	0x0	FIFO receive underflow 1 = host attempts to read from the I2C data register more times than the value of the transfer size register plus one
TX_OVF (IXR_TX_OVR)	6	wtc	0x0	FIFO transmit overflow 1 = host attempts to write to the I2C data register more times than the FIFO depth
RX_OVF (IXR_RX_OVR)	5	wtc	0x0	Receive overflow 1 = This bit is set whenever FIFO is full and a new byte is received. The new byte is not acknowledged and contents of the FIFO remains unchanged.
SLV_RDY (IXR_SLV_RDY)	4	wtc	0x0	Monitored slave ready 1 = addressed slave returns ACK.
TO (IXR_TO)	3	wtc	0x0	Transfer time out 1 = I2C sclk line is kept low for longer time
NACK (IXR_NACK)	2	wtc	0x0	Transfer not acknowledged 1 = slave responds with a NACK or master terminates the transfer before all data is supplied

Field Name	Bits	Type	Reset Value	Description
DATA (IXR_DATA)	1	wtc	0x0	More data 1 = Data being sent or received
COMP (IXR_COMP)	0	wtc	0x0	Transfer complete 1 = transfer is complete

Register (IIC) Transfer_size_reg0

Name	Transfer_size_reg0
Software Name	TRANS_SIZE
Relative Address	0x00000014
Absolute Address	i2c0: 0xE0004014 i2c1: 0xE0005014
Width	8 bits
Access Type	rw
Reset Value	0x00000000
Description	Transfer Size Register

Register Transfer_size_reg0 Details

This register's meaning varies according to the operating mode as follows:

- * Master transmitter mode: number of data bytes still not transmitted minus one
- * Master receiver mode: number of data bytes that are still expected to be received
- * Slave transmitter mode: number of bytes remaining in the FIFO after the master terminates the transfer
- * Slave receiver mode: number of valid data bytes in the FIFO

This register is cleared if CLR_FIFO bit in the control register is set.

Field Name	Bits	Type	Reset Value	Description
Transfer_Size (MASK)	7:0	rw	0x0	Transfer Size 0-255

Register (IIC) Slave_mon_pause_reg0

Name	Slave_mon_pause_reg0
Software Name	SLV_PAUSE
Relative Address	0x00000018

Absolute Address i2c0: 0xE0004018
 i2c1: 0xE0005018
 Width 8 bits
 Access Type mixed
 Reset Value 0x00000000
 Description Slave Monitor Pause Register

Register Slave_mon_pause_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	7:4	ro	0x0	Reserved, read as zero, ignored on write.
Pause (MASK)	3:0	rw	0x0	pause interval 0 - 7: pause interval

Register (IIC) Time_out_reg0

Name Time_out_reg0
 Software Name TIME_OUT
 Relative Address 0x0000001C
 Absolute Address i2c0: 0xE000401C
 i2c1: 0xE000501C
 Width 8 bits
 Access Type rw
 Reset Value 0x0000001F
 Description Time out register

Register Time_out_reg0 Details

Field Name	Bits	Type	Reset Value	Description
TO (MASK)	7:0	rw	0x1F	Time Out 127 - 32 : value of time out register

Register (IIC) Intrpt_mask_reg0

Name Intrpt_mask_reg0
 Software Name IMR
 Relative Address 0x00000020
 Absolute Address i2c0: 0xE0004020
 i2c1: 0xE0005020

Width	16 bits
Access Type	ro
Reset Value	0x000002FF
Description	Interrupt mask register

Register Intrpt_mask_reg0 Details

Each bit in this register corresponds to a bit in the interrupt status register. If bit *i* in the interrupt mask register is set, the corresponding bit in the interrupt status register is ignored. Otherwise, an interrupt is generated whenever bit *i* in the interrupt status register is set.

Bits in this register are set through a write to the interrupt disable register and are cleared through a write to the interrupt enable register.

All mask bits are set and all interrupts are disabled after reset.

Interrupt mask register has the same format as the interrupt status register.

Field Name	Bits	Type	Reset Value	Description
reserved	15:10	ro	0x0	Reserved, read as zero, ignored on write.
ARB_LOST (IXR_ARB_LOST)	9	ro	0x1	arbitration lost 1 = Mask this interrupt 0 = unmask this interrupt
reserved	8	ro	0x0	Reserved, read as zero, ignored on write.
RX_UNF (IXR_RX_UNF)	7	ro	0x1	FIFO receive underflow 1 = Mask this interrupt 0 = unmask this interrupt
TX_OVF (IXR_TX_OVR)	6	ro	0x1	FIFO transmit overflow 1 = Mask this interrupt 0 = unmask this interrupt
RX_OVF (IXR_RX_OVR)	5	ro	0x1	Receive overflow 1 = Mask this interrupt 0 = unmask this interrupt
SLV_RDY (IXR_SLV_RDY)	4	ro	0x1	Monitored slave ready 1 = Mask this interrupt 0 = unmask this interrupt
TO (IXR_TO)	3	ro	0x1	Transfer time out 1 = Mask this interrupt 0 = unmask this interrupt
NACK (IXR_NACK)	2	ro	0x1	Transfer not acknowledged 1 = Mask this interrupt 0 = unmask this interrupt

Field Name	Bits	Type	Reset Value	Description
DATA (IXR_DATA)	1	ro	0x1	More data 1 = Mask this interrupt 0 = unmask this interrupt
COMP (IXR_COMP)	0	ro	0x1	Transfer complete 1 = Mask this interrupt 0 = unmask this interrupt

Register (IIC) Intrpt_enable_reg0

Name	Intrpt_enable_reg0
Software Name	IER
Relative Address	0x00000024
Absolute Address	i2c0: 0xE0004024 i2c1: 0xE0005024
Width	16 bits
Access Type	mixed
Reset Value	0x00000000
Description	Interrupt Enable Register

Register Intrpt_enable_reg0 Details

This register has the same format as the interrupt status register.

Setting a bit in the interrupt enable register clears the corresponding mask bit in the interrupt mask register, effectively enabling corresponding interrupt to be generated.

Field Name	Bits	Type	Reset Value	Description
reserved	15:10	ro	0x0	Reserved, read as zero, ignored on write.
ARB_LOST (IXR_ARB_LOST)	9	wo	0x0	arbitration lost 1 = enable this interrupt
reserved	8	ro	0x0	Reserved, read as zero, ignored on write.
RX_UNF (IXR_RX_UNF)	7	wo	0x0	FIFO receive underflow 1 = enable this interrupt
TX_OVF (IXR_TX_OVR)	6	wo	0x0	FIFO transmit overflow 1 = enable this interrupt
RX_OVF (IXR_RX_OVR)	5	wo	0x0	Receive overflow 1 = enable this interrupt
SLV_RDY (IXR_SLV_RDY)	4	wo	0x0	Monitored slave ready 1 = enable this interrupt

Field Name	Bits	Type	Reset Value	Description
TO (IXR_TO)	3	wo	0x0	Transfer time out 1 = enable this interrupt
NACK (IXR_NACK)	2	wo	0x0	Transfer not acknowledged 1 = enable this interrupt
DATA (IXR_DATA)	1	wo	0x0	More data 1 = enable this interrupt
COMP (IXR_COMP)	0	wo	0x0	Transfer complete Will be set when transfer is complete 1 = enable this interrupt

Register (IIC) Intrpt_disable_reg0

Name	Intrpt_disable_reg0
Software Name	IDR
Relative Address	0x00000028
Absolute Address	i2c0: 0xE0004028 i2c1: 0xE0005028
Width	16 bits
Access Type	mixed
Reset Value	0x00000000
Description	Interrupt Disable Register

Register Intrpt_disable_reg0 Details

This register has the same format as the interrupt status register.

Setting a bit in the interrupt disable register sets the corresponding mask bit in the interrupt mask register, effectively disabling corresponding interrupt to be generated.

Field Name	Bits	Type	Reset Value	Description
reserved	15:10	ro	0x0	Reserved, read as zero, ignored on write.
ARB_LOST (IXR_ARB_LOST)	9	wo	0x0	arbitration lost 1 = disable this interrupt
reserved	8	ro	0x0	Reserved, read as zero, ignored on write.
RX_UNF (IXR_RX_UNF)	7	wo	0x0	FIFO receive underflow 1 = disable this interrupt
TX_OVF (IXR_TX_OVR)	6	wo	0x0	FIFO transmit overflow 1 = disable this interrupt
RX_OVF (IXR_RX_OVR)	5	wo	0x0	Receive overflow 1 = disable this interrupt

Field Name	Bits	Type	Reset Value	Description
SLV_RDY (IXR_SLV_RDY)	4	wo	0x0	Monitored slave ready 1 = disable this interrupt
TO (IXR_TO)	3	wo	0x0	Transfer time out 1 = disable this interrupt
NACK (IXR_NACK)	2	wo	0x0	Transfer not acknowledged 1 = disable this interrupt
DATA (IXR_DATA)	1	wo	0x0	Master Write or Slave Transmitter Master Read or Slave Receiver 1 = disable this interrupt
COMP (IXR_COMP)	0	wo	0x0	Transfer complete Will be set when transfer is complete 1 = disable this interrupt

B.23 L2 Cache (L2Cpl310)

Module Name	L2 Cache (L2Cpl310)
Base Address	0xF8F02000 l2cache
Description	L2 cache PL310
Version	r3p2v
Doc Version	1.0
Vendor Info	ARM

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
reg0_cache_id	0x00000000	32	mixed	0x410000C8	cache ID register, Returns the 32-bit device ID code it reads off the CACHEID input bus. The value is specified by the system integrator. Reset value: 0x410000c8
reg0_cache_type	0x00000004	32	mixed	0x9E300300	cache type register, Returns the 32-bit cache type. Reset value: 0x1c100100
reg1_control	0x00000100	32	mixed	0x00000000	control register, reset value: 0x0
reg1_aux_control	0x00000104	32	mixed	0x02050000	auxiliary control register, reset value: 0x02020000+H273
reg1_tag_ram_control	0x00000108	32	mixed	0x00000777	Configures Tag RAM latencies
reg1_data_ram_control	0x0000010C	32	mixed	0x00000777	configures data RAM latencies
reg2_ev_counter_ctrl	0x00000200	32	mixed	0x00000000	Permits the event counters to be enabled and reset.
reg2_ev_counter1_cfg	0x00000204	32	mixed	0x00000000	Enables event counter 1 to be driven by a specific event. Counter 1 increments when the event occurs.
reg2_ev_counter0_cfg	0x00000208	32	mixed	0x00000000	Enables event counter 0 to be driven by a specific event. Counter 0 increments when the event occurs.

Register Name	Address	Width	Type	Reset Value	Description
reg2_ev_counter1	0x0000020C	32	rw	0x00000000	Enable the programmer to read off the counter value. The counter counts an event as specified by the Counter Configuration Registers. The counter can be preloaded if counting is disabled and reset by the Event Counter Control Register.
reg2_ev_counter0	0x00000210	32	rw	0x00000000	Enable the programmer to read off the counter value. The counter counts an event as specified by the Counter Configuration Registers. The counter can be preloaded if counting is disabled and reset by the Event Counter Control Register.
reg2_int_mask	0x00000214	32	mixed	0x00000000	This register enables or masks interrupts from being triggered on the external pins of the cache controller. Figure 3-8 on page 3-17 shows the register bit assignments. The bit assignments enables the masking of the interrupts on both their individual outputs and the combined L2CCINTR line. Clearing a bit by writing a 0, disables the interrupt triggering on that pin. All bits are cleared by a reset. You must write to the register bits with a 1 to enable the generation of interrupts. 1 = Enabled. 0 = Masked. This is the default.

Register Name	Address	Width	Type	Reset Value	Description
reg2_int_mask_status	0x00000218	32	mixed	0x00000000	<p>This register is a read-only. It returns the masked interrupt status. This register can be accessed by secure and non-secure operations. The register gives an AND function of the raw interrupt status with the values of the interrupt mask register. All the bits are cleared by a reset. A write to this register is ignored. Bits read can be HIGH or LOW:</p> <p>HIGH If the bits read HIGH, they reflect the status of the input lines triggering an interrupt.</p> <p>LOW If the bits read LOW, either no interrupt has been generated, or the interrupt is masked.</p>
reg2_int_raw_status	0x0000021C	32	mixed	0x00000000	<p>The Raw Interrupt Status Register enables the interrupt status that excludes the masking logic.</p> <p>Bits read can be HIGH or LOW:</p> <p>HIGH If the bits read HIGH, they reflect the status of the input lines triggering an interrupt.</p> <p>LOW If the bits read LOW, no interrupt has been generated.</p>
reg2_int_clear	0x00000220	32	mixed	0x00000000	<p>Clears the Raw Interrupt Status Register bits.</p> <p>When a bit is written as 1, it clears the corresponding bit in the Raw Interrupt Status Register. When a bit is written as 0, it has no effect</p>
reg7_cache_sync	0x00000730	32	mixed	0x00000000	<p>Drain the STB. Operation complete when all buffers, LRB, LFB, STB, and EB, are empty</p>
reg7_inv_pa	0x00000770	32	mixed	0x00000000	<p>Invalidate Line by PA: Specific L2 cache line is marked as not valid</p>

Register Name	Address	Width	Type	Reset Value	Description
reg7_inv_way	0x0000077C	32	mixed	0x00000000	Invalidate by Way Invalidate all data in specified ways, including dirty data. An Invalidate by way while selecting all cache ways is equivalent to invalidating all cache entries. Completes as a background task with the way, or ways, locked, preventing allocation.
reg7_clean_pa	0x000007B0	32	mixed	0x00000000	Clean Line by PA Write the specific L2 cache line to L3 main memory if the line is marked as valid and dirty. The line is marked as not dirty. The valid bit is unchanged
reg7_clean_index	0x000007B8	32	mixed	0x00000000	Clean Line by Set/Way Write the specific L2 cache line within the specified way to L3 main memory if the line is marked as valid and dirty. The line is marked as not dirty. The valid bit is unchanged
reg7_clean_way	0x000007BC	32	mixed	0x00000000	Clean by Way Writes each line of the specified L2 cache ways to L3 main memory if the line is marked as valid and dirty. The lines are marked as not dirty. The valid bits are unchanged. Completes as a background task with the way, or ways, locked, preventing allocation.
reg7_clean_inv_pa	0x000007F0	32	mixed	0x00000000	Clean and Invalidate Line by PA Write the specific L2 cache line to L3 main memory if the line is marked as valid and dirty. The line is marked as not valid
reg7_clean_inv_index	0x000007F8	32	mixed	0x00000000	Clean and Invalidate Line by Set/Way Write the specific L2 cache line within the specified way to L3 main memory if the line is marked as valid and dirty. The line is marked as not valid

Register Name	Address	Width	Type	Reset Value	Description
reg7_clean_inv_way	0x000007FC	32	mixed	0x00000000	Clean and Invalidate by Way Writes each line of the specified L2 cache ways to L3 main memory if the line is marked as valid and dirty. The lines are marked as not valid. Completes as a background task with the way, or ways, locked, preventing allocation.
reg9_d_lockdown0	0x00000900	32	mixed	0x00000000	These registers can prevent new addresses from being allocated and can also prevent data from being evicted out of the L2 cache. Such behavior can distinguish instructions from data transactions. To control the cache lockdown by way and the cache lockdown by master mechanisms see the tables from Table 3-20 to Table 3-35 on page 3-31. For these tables each bit has the following meaning: 0 allocation can occur in the corresponding way. 1 there is no allocation in the corresponding way.
reg9_i_lockdown0	0x00000904	32	mixed	0x00000000	instruction lock down 0
reg9_d_lockdown1	0x00000908	32	mixed	0x00000000	data lock down 1
reg9_i_lockdown1	0x0000090C	32	mixed	0x00000000	instruction lock down 1
reg9_d_lockdown2	0x00000910	32	mixed	0x00000000	data lock down 2
reg9_i_lockdown2	0x00000914	32	mixed	0x00000000	instruction lock down 2
reg9_d_lockdown3	0x00000918	32	mixed	0x00000000	data lock down 3
reg9_i_lockdown3	0x0000091C	32	mixed	0x00000000	instruction lock down 3
reg9_d_lockdown4	0x00000920	32	mixed	0x00000000	data lock down 4
reg9_i_lockdown4	0x00000924	32	mixed	0x00000000	instruction lock down 4
reg9_d_lockdown5	0x00000928	32	mixed	0x00000000	data lock down 5
reg9_i_lockdown5	0x0000092C	32	mixed	0x00000000	instruction lock down 5
reg9_d_lockdown6	0x00000930	32	mixed	0x00000000	data lock down 6
reg9_i_lockdown6	0x00000934	32	mixed	0x00000000	instruction lock down 6
reg9_d_lockdown7	0x00000938	32	mixed	0x00000000	data lock down 7

Register Name	Address	Width	Type	Reset Value	Description
reg9_i_lockdown7	0x0000093C	32	mixed	0x00000000	instruction lock down 7
reg9_lock_line_en	0x00000950	32	mixed	0x00000000	Lockdown by Line Enable Register.
reg9_unlock_way	0x00000954	32	mixed	0x00000000	Cache lockdown by way To control the cache lockdown by way and the cache lockdown by master mechanisms see the tables from Table 3-20 to Table 3-35 on page 3-31. For these tables each bit has the following meaning: 0 allocation can occur in the corresponding way. 1 there is no allocation in the corresponding way.
reg12_addr_filtering_start	0x00000C00	32	mixed	0x40000001	When two masters are implemented, you can redirect a whole address range to master 1 (M1). When address_filtering_enable is set, all accesses with address \geq address_filtering_start and $<$ address_filtering_end are automatically directed to M1. All other accesses are directed to M0. This feature is programmed using two registers.
reg12_addr_filtering_end	0x00000C04	32	mixed	0xFFF00000	When two masters are implemented, you can redirect a whole address range to master 1 (M1). When address_filtering_enable is set, all accesses with address \geq address_filtering_start and $<$ address_filtering_end are automatically directed to M1. All other accesses are directed to M0. This feature is programmed using two registers.

Register Name	Address	Width	Type	Reset Value	Description
reg15_debug_ctrl	0x00000F40	32	mixed	0x00000000	The Debug Control Register forces specific cache behavior required for debug. This register has read-only, non-secure, or read and write, secure, permission. Any secure access and non-secure access can read this register. Only a secure access can write to this register. If a non-secure access tries to write to this register the register issues a DECERR response and does not update.
reg15_prefetch_ctrl	0x00000F60	32	mixed	0x00000000	Purpose Enables prefetch-related features that can improve system performance. Usage constraints This register has both read-only, non-secure, and read and write, secure, permissions. Any secure or non-secure access can read this register. Only a secure access can write to this register. If a non-secure access attempts to write to this register, the register
reg15_power_ctrl	0x00000F80	32	mixed	0x00000000	Purpose Controls the operating mode clock and power modes. Usage constraints There are no usage constraints.

Register ([L2Cpl310](#)) reg0_cache_id

Name	reg0_cache_id
Relative Address	0x00000000
Absolute Address	0xF8F02000
Width	32 bits
Access Type	mixed
Reset Value	0x410000C8

Description cache ID register, Returns the 32-bit device ID code it reads off the CACHEID input bus.
The value is specified by the system integrator. Reset value: 0x410000c8

Register reg0_cache_id Details

Field Name	Bits	Type	Reset Value	Description
implementer	31:24	ro	0x41	0x41, ARM
reserved	23:16	waz	0x0	reserved
cache_id	15:10	ro	0x0	cache id
part_num	9:6	ro	0x3	part number
rtl_release	5:0	ro	0x8	RTL release R3p2

Register ([L2Cpl310](#)) reg0_cache_type

Name reg0_cache_type
Relative Address 0x00000004
Absolute Address 0xF8F02004
Width 32 bits
Access Type mixed
Reset Value 0x9E300300
Description cache type register, Returns the 32-bit cache type. Reset value: 0x1c100100

Register reg0_cache_type Details

Field Name	Bits	Type	Reset Value	Description
data_banking	31	ro	0x1	0 = Data banking not implemented. 1 = Data banking implemented.
reserved	30:29	waz	0x0	reserved
ctype	28:25	ro	0xF	11xy, where: x=1 if pl310_LOCKDOWN_BY_MASTER is defined, otherwise 0 y=1 if pl310_LOCKDOWN_BY_LINE is defined, otherwise 0.
H	24	ro	0x0	unified
Dsize_23	23	waz,r az	0x0	fixed to 0
Dsize_mid	22:20	ro	0x3	L2 cache way size Read from Auxiliary Control Register 19 through 17
Dsize_19	19	waz,r az	0x0	fixed to 0

Field Name	Bits	Type	Reset Value	Description
L2_assoc_D	18	ro	0x0	Read from Auxiliary Control Register bit 16
reserved	17:14	waz	0x0	reserved
l2cache_line_len_D	13:12	ro	0x0	L2 cache line length - 00-32 bytes
Isize_11	11	waz,r az	0x0	fixed to 0
Isize_mid	10:8	ro	0x3	L2 cache way size Read from Auxiliary Control Register[19:17]
Isize_7	7	waz,r az	0x0	fixed to 0
L2_assoc_I	6	ro	0x0	Read from Auxiliary Control Register bit 16
reserved	5:2	waz	0x0	reserved
l2cache_line_len_I	1:0	ro	0x0	L2 cache line length - 00-32 bytes

Register ([L2Cpl310](#)) reg1_control

Name	reg1_control
Relative Address	0x00000100
Absolute Address	0xF8F02100
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	control register, reset value: 0x0

Register reg1_control Details

Field Name	Bits	Type	Reset Value	Description
reserved	30:1	waz,r az	0x0	reserved, reserved
l2_enable	0	rw	0x0	0 = L2 Cache is disabled. This is the default value. 1 = L2 Cache is enabled.

Register ([L2Cpl310](#)) reg1_aux_control

Name	reg1_aux_control
Relative Address	0x00000104
Absolute Address	0xF8F02104
Width	32 bits

Access Type mixed

Reset Value 0x02050000

Description auxiliary control register, reset value: 0x02020000+H273

Register reg1_aux_control Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	waz,r az	0x0	reserved, reserved
early_bresp_en	30	rw	0x0	Early BRESP enable 0 = Early BRESP disabled. This is the default. 1 = Early BRESP enabled.
instr_prefetch_en	29	rw	0x0	Instruction prefetch enable 0 = Instruction prefetching disabled. This is the default. 1 = Instruction prefetching enabled.
data_prefetch_en	28	rw	0x0	Data prefetch enable 0 = Data prefetching disabled. This is the default. 1 = Data prefetching enabled.
nonsec_inte_access_ctrl	27	rw	0x0	Non-secure interrupt access control 0 = Interrupt Clear, 0x220, and Interrupt Mask, 0x214, can only be modified or read with secure accesses. This is the default. 1 = Interrupt Clear, 0x220, and Interrupt Mask, 0x214, can be modified or read with secure or non-secure accesses.
nonsec_lockdown_en	26	rw	0x0	Non-secure lockdown enable 0 = Lockdown registers cannot be modified using non-secure accesses. This is the default. 1 = Non-secure accesses can write to the lockdown registers.
cache_replace_policy	25	rw	0x1	Cache replacement policy 0 = pseudo-random replacement using lfsr. 1 = round-robin replacement. This is the default.

Field Name	Bits	Type	Reset Value	Description
force_write_alloc	24:23	rw	0x0	Force write allocate b00 = Use AWCACHE attributes for WA. This is the default. b01 = Force no allocate, set WA bit always 0. b10 = Override AWCACHE attributes, set WA bit always 1, all cacheable write misses become write allocated. b11 = Internally mapped to 00. See Cache operation on page 2-11 for more information.
shared_attr_override_en	22	rw	0x0	Shared attribute override enable 0 = Treats shared accesses as specified in Shareable attribute on page 2-15. This is the default. 1 = Shared attribute internally ignored.
parity_en	21	rw	0x0	Parity enable 0 = Disabled. This is the default. 1 = Enabled
event_mon_bus_en	20	rw	0x0	Event monitor bus enable 0 = Disabled. This is the default. 1 = Enabled
way_size	19:17	rw	0x2	Way-size b000 = Reserved, internally mapped to 16KB. b001 = 16KB. b010 = 32KB. b011 = 64KB. b100 = 128KB. b101 = 256KB. b110 = 512KB. b111 = Reserved, internally mapped to 512 KB.
associativity	16	rw	0x1	Associativity 0 = 8-way. 1 = 16-way.
reserved	15:14	waz,raz	0x0	reserved, reserved
shared_attr_inva_en	13	rw	0x0	Shared Attribute Invalidate Enable 0 = Shared invalidate behavior disabled. This is the default. 1 = Shared invalidate behavior enabled, if Shared Attribute Override Enable bit not set. See Shareable attribute on page 2-15.

Field Name	Bits	Type	Reset Value	Description
ex_cache_config	12	rw	0x0	Exclusive cache configuration 0 = Disabled. This is the default. 1 = Enabled,
store_buff_dev_lim_en	11	rw	0x0	Store buffer device limitation Enable 0 = Store buffer device limitation disabled. Device writes can take all slots in store buffer. This is the default. 1= Store buffer device limitation enabled. Device writes cannot take all slots in store buffer when connected to the Cortex-A9 MPCore processor. There is always one available slot to service Normal Memory
high_pr_so_dev_rd_en	10	rw	0x0	High Priority for SO and Dev Reads Enable 0 = Strongly Ordered and Device reads have lower priority than cacheable accesses when arbitrated in the L2CC (L2C-310) master ports. This is the default. 1 = Strongly Ordered and Device reads get the highest priority when arbitrated in the L2CC (L2C-310) master ports.
reserved	9:1	waz,r az	0x0	reserved, reserved
full_line_zero_enable	0	rw	0x0	Full Line of Zero Enable 0 = Full line of write zero behavior disabled. This is the default. 1 = Full line of write zero behavior Enabled.

Register ([L2Cpl310](#)) reg1_tag_ram_control

Name	reg1_tag_ram_control
Relative Address	0x00000108
Absolute Address	0xF8F02108
Width	32 bits
Access Type	mixed
Reset Value	0x00000777
Description	Configures Tag RAM latencies

Register reg1_tag_ram_control Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:11	waz,r az	0x0	reserved, reserved
ram_wr_access_lat	10:8	rw	0x7	RAM write access latency Default value depends on the value of pl310_TAG_WRITE_LAT b000 = 1 cycle of latency, there is no additional latency. b001 = 2 cycles of latency. b010 = 3 cycles of latency. b011 = 4 cycles of latency. b100 = 5 cycles of latency. b101 = 6 cycles of latency. b110 = 7 cycles of latency. b111 = 8 cycles of latency
reserved	7	waz,r az	0x0	reserved, reserved
ram_rd_access_lat	6:4	rw	0x7	RAM read access latency Default value depends on the value of pl310_TAG_READ_LAT b000 = 1 cycle of latency, there is no additional latency. b001 = 2 cycles of latency. b010 = 3 cycles of latency. b011 = 4 cycles of latency. b100 = 5 cycles of latency. b101 = 6 cycles of latency. b110 = 7 cycles of latency. b111 = 8 cycles of latency.
reserved	3	waz,r az	0x0	reserved, reserved
ram_setup_lat	2:0	rw	0x7	RAM setup latency Default value depends on the value of pl310_TAG_SETUP_LAT b000 = 1 cycle of latency, there is no additional latency. b001 = 2 cycles of latency. b010 = 3 cycles of latency. b011 = 4 cycles of latency. b100 = 5 cycles of latency. b101 = 6 cycles of latency. b110 = 7 cycles of latency. b111 = 8 cycles of latency.

Register ([L2Cpl310](#)) reg1_data_ram_control

Name	reg1_data_ram_control
Relative Address	0x0000010C
Absolute Address	0xF8F0210C
Width	32 bits
Access Type	mixed
Reset Value	0x00000777
Description	configures data RAM latencies

Register reg1_data_ram_control Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:11	waz,r az	0x0	reserved, reserved
ram_wr_access_lat	10:8	rw	0x7	RAM write access latency Default value depends on the value of pl310_DATA_WRITE_LAT b000 = 1 cycle of latency, there is no additional latency. b001 = 2 cycles of latency. b010 = 3 cycles of latency. b011 = 4 cycles of latency. b100 = 5 cycles of latency. b101 = 6 cycles of latency. b110 = 7 cycles of latency. b111 = 8 cycles of latency
reserved	7	waz,r az	0x0	reserved, reserved
ram_rd_access_lat	6:4	rw	0x7	RAM read access latency Default value depends on the value of pl310_DATA_READ_LAT b000 = 1 cycle of latency, there is no additional latency. b001 = 2 cycles of latency. b010 = 3 cycles of latency. b011 = 4 cycles of latency. b100 = 5 cycles of latency. b101 = 6 cycles of latency. b110 = 7 cycles of latency. b111 = 8 cycles of latency.

Field Name	Bits	Type	Reset Value	Description
reserved	3	waz,r az	0x0	reserved, reserved
ran_setup_lat	2:0	rw	0x7	RAM setup latency Default value depends on the value of pl310_DATA_SETUP_LAT b000 = 1 cycle of latency, there is no additional latency. b001 = 2 cycles of latency. b010 = 3 cycles of latency. b011 = 4 cycles of latency. b100 = 5 cycles of latency. b101 = 6 cycles of latency. b110 = 7 cycles of latency. b111 = 8 cycles of latency.

Register ([L2Cpl310](#)) reg2_ev_counter_ctrl

Name	reg2_ev_counter_ctrl
Relative Address	0x00000200
Absolute Address	0xF8F02200
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Permits the event counters to be enabled and reset.

Register reg2_ev_counter_ctrl Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:3	waz,r az	0x0	reserved, reserved
counter_reset	2:1	raz,r w	0x0	Always Read as zero. The following counters are reset when a 1 is written to the following bits: bit[2] = Event Counter1 reset bit[1] = Event Counter0 reset.
ev_ctr_en	0	rw	0x0	Event counter enable 0 = Event Counting Disable. This is the default. 1 = Event Counting Enable.

Register ([L2Cpl310](#)) reg2_ev_counter1_cfg

Name	reg2_ev_counter1_cfg
------	----------------------

Relative Address	0x00000204
Absolute Address	0xF8F02204
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Enables event counter 1 to be driven by a specific event. Counter 1 increments when the event occurs.

Register reg2_ev_counter1_cfg Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:6	waz,r az	0x0	reserved, reserved
ctr_ev_src	5:2	rw	0x0	Counter event source Event Encoding Counter Disabled b0000 CO b0001 DRHIT b0010 DRREQ b0011 DWHIT b0100 DWREQ b0101 DWTREQ b0110 IRHIT b0111 IRREQ b1000 WA b1001 IPFALLOCC b1010 EPFHIT b1011 EPFALLOCC b1100 SRRCVDC b1101 SRCONF b1110 EPFRCVDC b1111
ev_ctr_intr_gen	1:0	rw	0x0	Event counter interrupt generation b00 = Disabled. This is the default. b01 = Enabled: Increment condition. b10 = Enabled: Overflow condition. b11 = Interrupt generation is disabled.

Register ([L2Cpl310](#)) reg2_ev_counter0_cfg

Name	reg2_ev_counter0_cfg
Relative Address	0x00000208
Absolute Address	0xF8F02208

Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Enables event counter 0 to be driven by a specific event. Counter 0 increments when the event occurs.

Register reg2_ev_counter0_cfg Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:6	waz,raz	0x0	reserved, reserved
ctr_ev_src	5:2	rw	0x0	Counter event source Event Encoding Counter Disabled b0000 CO b0001 DRHIT b0010 DRREQ b0011 DWHIT b0100 DWREQ b0101 DWTREQ b0110 IRHIT b0111 IRREQ b1000 WA b1001 IPFALLOCC b1010 EPFHIT b1011 EPFALLOCC b1100 SRRCVDC b1101 SRCONFC b1110 EPFRCVDC b1111
ev_ctr_intr_gen	1:0	rw	0x0	Event counter interrupt generation b00 = Disabled. This is the default. b01 = Enabled: Increment condition. b10 = Enabled: Overflow condition. b11 = Interrupt generation is disabled.

Register ([L2Cpl310](#)) reg2_ev_counter1

Name	reg2_ev_counter1
Relative Address	0x0000020C
Absolute Address	0xF8F0220C
Width	32 bits
Access Type	rw
Reset Value	0x00000000

Description Enable the programmer to read off the counter value. The counter counts an event as specified by the Counter Configuration Registers. The counter can be preloaded if counting is disabled and reset by the Event Counter Control Register.

Register reg2_ev_counter1 Details

Field Name	Bits	Type	Reset Value	Description
counter_val	31:0	rw	0x0	Total of the event selected. If a counter reaches its maximum value, it saturates at that value until it is reset.

Register ([L2Cpl310](#)) reg2_ev_counter0

Name reg2_ev_counter0

Relative Address 0x00000210

Absolute Address 0xF8F02210

Width 32 bits

Access Type rw

Reset Value 0x00000000

Description Enable the programmer to read off the counter value. The counter counts an event as specified by the Counter Configuration Registers. The counter can be preloaded if counting is disabled and reset by the Event Counter Control Register.

Register reg2_ev_counter0 Details

Field Name	Bits	Type	Reset Value	Description
counter_val	31:0	rw	0x0	Total of the event selected. If a counter reaches its maximum value, it saturates at that value until it is reset.

Register ([L2Cpl310](#)) reg2_int_mask

Name reg2_int_mask

Relative Address 0x00000214

Absolute Address 0xF8F02214

Width 32 bits

Access Type mixed

Reset Value 0x00000000

Description This register enables or masks interrupts from being triggered on the external pins of the cache controller. Figure 3-8 on page 3-17 shows the register bit assignments. The bit assignments enables the masking of the interrupts on both their individual outputs and the combined L2CCINTR line. Clearing a bit by writing a 0, disables the interrupt triggering on that pin. All bits are cleared by a reset. You must write to the register bits with a 1 to enable the generation of interrupts.

1 = Enabled.

0 = Masked. This is the default.

Register reg2_int_mask Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:9	waz,r az	0x0	reserved, reserved
DECERR	8	rw	0x0	DECERR: DECERR from L3
SLVERR	7	rw	0x0	SLVERR: SLVERR from L3
ERRRD	6	rw	0x0	ERRRD: Error on L2 data RAM, Read
ERRRT	5	rw	0x0	ERRRT: Error on L2 tag RAM, Read
ERRWD	4	rw	0x0	ERRWD: Error on L2 data RAM, Write
ERRWT	3	rw	0x0	ERRWT: Error on L2 tag RAM, Write
PARRD	2	rw	0x0	PARRD: Parity Error on L2 data RAM, Read
PARRT	1	rw	0x0	PARRT: Parity Error on L2 tag RAM, Read
ECNTR	0	rw	0x0	ECNTR: Event Counter1/0 Overflow Increment

Register ([L2Cpl310](#)) reg2_int_mask_status

Name	reg2_int_mask_status
Relative Address	0x00000218
Absolute Address	0xF8F02218
Width	32 bits
Access Type	mixed
Reset Value	0x00000000

Description

This register is a read-only. It returns the masked interrupt status. This register can be accessed by secure and non-secure operations. The register gives an AND function of the raw interrupt status with the values of the interrupt mask register. All the bits are cleared by a reset. A write to this register is ignored. Bits read can be HIGH or LOW:

HIGH If the bits read HIGH, they reflect the status of the input lines triggering an interrupt.

LOW If the bits read LOW, either no interrupt has been generated, or the interrupt is masked.

Register reg2_int_mask_status Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:9	raz	0x0	reserved
DECERR	8	ro	0x0	DECERR: DECERR from L3
SLVERR	7	ro	0x0	SLVERR: SLVERR from L3
ERRRD	6	ro	0x0	ERRRD: Error on L2 data RAM, Read
ERRRT	5	ro	0x0	ERRRT: Error on L2 tag RAM, Read
ERRWD	4	ro	0x0	ERRWD: Error on L2 data RAM, Write
ERRWT	3	ro	0x0	ERRWT: Error on L2 tag RAM, Write
PARRD	2	ro	0x0	PARRD: Parity Error on L2 data RAM, Read
PARRT	1	ro	0x0	PARRT: Parity Error on L2 tag RAM, Read
ECNTR	0	ro	0x0	ECNTR: Event Counter1/0 Overflow Increment

Register ([L2Cpl310](#)) reg2_int_raw_status

Name reg2_int_raw_status

Relative Address 0x0000021C

Absolute Address 0xF8F0221C

Width 32 bits

Access Type mixed

Reset Value 0x00000000

Description

The Raw Interrupt Status Register enables the interrupt status that excludes the masking logic.

Bits read can be HIGH or LOW:

HIGH If the bits read HIGH, they reflect the status of the input lines triggering an interrupt.

LOW If the bits read LOW, no interrupt has been generated.

Register reg2_int_raw_status Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:9	raz	0x0	reserved
DECERR	8	ro	0x0	DECERR: DECERR from L3
SLVERR	7	ro	0x0	SLVERR: SLVERR from L3
ERRRD	6	ro	0x0	ERRRD: Error on L2 data RAM, Read
ERRRT	5	ro	0x0	ERRRT: Error on L2 tag RAM, Read
ERRWD	4	ro	0x0	ERRWD: Error on L2 data RAM, Write
ERRWT	3	ro	0x0	ERRWT: Error on L2 tag RAM, Write
PARRD	2	ro	0x0	PARRD: Parity Error on L2 data RAM, Read
PARRT	1	ro	0x0	PARRT: Parity Error on L2 tag RAM, Read
ECNTR	0	ro	0x0	ECNTR: Event Counter1/0 Overflow Increment

Register ([L2Cpl310](#)) reg2_int_clear

Name	reg2_int_clear
Relative Address	0x00000220
Absolute Address	0xF8F02220
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	<p>Clears the Raw Interrupt Status Register bits.</p> <p>When a bit is written as 1, it clears the corresponding bit in the Raw Interrupt Status Register. When a bit is written as 0, it has no effect</p>

Register reg2_int_clear Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:9	raz	0x0	reserved
DECERR	8	wtc	0x0	DECERR: DECERR from L3
SLVERR	7	wtc	0x0	SLVERR: SLVERR from L3
ERRRD	6	wtc	0x0	ERRRD: Error on L2 data RAM, Read
ERRRT	5	wtc	0x0	ERRRT: Error on L2 tag RAM, Read
ERRWD	4	wtc	0x0	ERRWD: Error on L2 data RAM, Write
ERRWT	3	wtc	0x0	ERRWT: Error on L2 tag RAM, Write
PARRD	2	wtc	0x0	PARRD: Parity Error on L2 data RAM, Read

Field Name	Bits	Type	Reset Value	Description
PARRT	1	wtc	0x0	PARRT: Parity Error on L2 tag RAM, Read
ECNTR	0	wtc	0x0	ECNTR: Event Counter1/0 Overflow Increment

Register ([L2Cpl310](#)) reg7_cache_sync

Name	reg7_cache_sync
Relative Address	0x00000730
Absolute Address	0xF8F02730
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Drain the STB. Operation complete when all buffers, LRB, LFB, STB, and EB, are empty

Register reg7_cache_sync Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	waz	0x0	reserved
c	0	rw	0x0	Cache Sync: Drain the STB. Operation complete when all buffers, LRB, LFB, STB, and EB, are empty.

Register ([L2Cpl310](#)) reg7_inv_pa

Name	reg7_inv_pa
Relative Address	0x00000770
Absolute Address	0xF8F02770
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Invalidate Line by PA: Specific L2 cache line is marked as not valid

Register reg7_inv_pa Details

Field Name	Bits	Type	Reset Value	Description
tag	31:12	rw	0x0	tag
index	11:5	rw	0x0	index

Field Name	Bits	Type	Reset Value	Description
reserved	4:1	waz	0x0	reserved
c	0	rw	0x0	C Flag When written must be 0. When read, indicates that a background operation is in progress

Register ([L2Cpl310](#)) reg7_inv_way

Name	reg7_inv_way
Relative Address	0x0000077C
Absolute Address	0xF8F0277C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Invalidate by Way Invalidate all data in specified ways, including dirty data. An Invalidate by way while selecting all cache ways is equivalent to invalidating all cache entries. Completes as a background task with the way, or ways, locked, preventing allocation.

Register reg7_inv_way Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	waz	0x0	reserved
way_bits	7:0	rw	0x0	way bits: You can select multiple ways at the same time, by setting the Way bits to 1

Register ([L2Cpl310](#)) reg7_clean_pa

Name	reg7_clean_pa
Relative Address	0x000007B0
Absolute Address	0xF8F027B0
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Clean Line by PA Write the specific L2 cache line to L3 main memory if the line is marked as valid and dirty. The line is marked as not dirty. The valid bit is unchanged

Register reg7_clean_pa Details

Field Name	Bits	Type	Reset Value	Description
tag	31:12	rw	0x0	tag
index	11:5	rw	0x0	index
reserved	4:1	waz	0x0	reserved
c	0	rw	0x0	C Flag When written must be 0. When read, indicates that a background operation is in progress

Register ([L2Cpl310](#)) reg7_clean_index

Name	reg7_clean_index
Relative Address	0x000007B8
Absolute Address	0xF8F027B8
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Clean Line by Set/Way Write the specific L2 cache line within the specified way to L3 main memory if the line is marked as valid and dirty. The line is marked as not dirty. The valid bit is unchanged

Register reg7_clean_index Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	waz	0x0	reserved
way	30:28	rw	0x0	way
reserved	27:12	waz	0x0	reserved
index	11:5	rw	0x0	index
reserved	4:1	waz	0x0	reserved
c	0	rw	0x0	c

Register ([L2Cpl310](#)) reg7_clean_way

Name	reg7_clean_way
Relative Address	0x000007BC
Absolute Address	0xF8F027BC
Width	32 bits

Access Type	mixed
Reset Value	0x00000000
Description	Clean by Way Writes each line of the specified L2 cache ways to L3 main memory if the line is marked as valid and dirty. The lines are marked as not dirty. The valid bits are unchanged. Completes as a background task with the way, or ways, locked, preventing allocation.

Register reg7_clean_way Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	waz	0x0	reserved
way_bits	7:0	rw	0x0	way bits: You can select multiple ways at the same time, by setting the Way bits to 1

Register ([L2Cpl310](#)) reg7_clean_inv_pa

Name	reg7_clean_inv_pa
Relative Address	0x000007F0
Absolute Address	0xF8F027F0
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Clean and Invalidate Line by PA Write the specific L2 cache line to L3 main memory if the line is marked as valid and dirty. The line is marked as not valid

Register reg7_clean_inv_pa Details

Field Name	Bits	Type	Reset Value	Description
tag	31:12	rw	0x0	tag
index	11:5	rw	0x0	index
reserved	4:1	waz	0x0	reserved
c	0	rw	0x0	C Flag When written must be 0. When read, indicates that a background operation is in progress

Register ([L2Cpl310](#)) reg7_clean_inv_index

Name	reg7_clean_inv_index
------	----------------------

Relative Address	0x000007F8
Absolute Address	0xF8F027F8
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Clean and Invalidate Line by Set/Way Write the specific L2 cache line within the specified way to L3 main memory if the line is marked as valid and dirty. The line is marked as not valid

Register reg7_clean_inv_index Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	waz	0x0	reserved
way	30:28	rw	0x0	way
reserved	27:12	waz	0x0	reserved
index	11:5	rw	0x0	index
reserved	4:1	waz	0x0	reserved
c	0	rw	0x0	c

Register ([L2Cpl310](#)) reg7_clean_inv_way

Name	reg7_clean_inv_way
Relative Address	0x000007FC
Absolute Address	0xF8F027FC
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Clean and Invalidate by Way Writes each line of the specified L2 cache ways to L3 main memory if the line is marked as valid and dirty. The lines are marked as not valid. Completes as a background task with the way, or ways, locked, preventing allocation.

Register reg7_clean_inv_way Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	waz	0x0	reserved
way_bits	7:0	rw	0x0	way bits: You can select multiple ways at the same time, by setting the Way bits to 1

Register ([L2Cpl310](#)) reg9_d_lockdown0

Name	reg9_d_lockdown0
Relative Address	0x00000900
Absolute Address	0xF8F02900
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	<p>These registers can prevent new addresses from being allocated and can also prevent data from being evicted out of the L2 cache. Such behavior can distinguish instructions from data transactions.</p> <p>To control the cache lockdown by way and the cache lockdown by master mechanisms see the tables from Table 3-20 to Table 3-35 on page 3-31. For these tables each bit has the following meaning:</p> <p>0 allocation can occur in the corresponding way.</p> <p>1 there is no allocation in the corresponding way.</p>

Register reg9_d_lockdown0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
DATALOCK000	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 000

Register ([L2Cpl310](#)) reg9_i_lockdown0

Name	reg9_i_lockdown0
Relative Address	0x00000904
Absolute Address	0xF8F02904
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	instruction lock down 0

Register reg9_i_lockdown0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
INSTRLOCK000	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 000

Register ([L2Cpl310](#)) reg9_d_lockdown1

Name	reg9_d_lockdown1
Relative Address	0x00000908
Absolute Address	0xF8F02908
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	data lock down 1

Register reg9_d_lockdown1 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
DATALOCK001	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 001

Register ([L2Cpl310](#)) reg9_i_lockdown1

Name	reg9_i_lockdown1
Relative Address	0x0000090C
Absolute Address	0xF8F0290C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	instruction lock down 1

Register reg9_i_lockdown1 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
INSTRLOCK001	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 001

Register ([L2Cpl310](#)) reg9_d_lockdown2

Name	reg9_d_lockdown2
Relative Address	0x00000910
Absolute Address	0xF8F02910
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	data lock down 2

Register reg9_d_lockdown2 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
DATALOCK010	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 010

Register ([L2Cpl310](#)) reg9_i_lockdown2

Name	reg9_i_lockdown2
Relative Address	0x00000914
Absolute Address	0xF8F02914
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	instruction lock down 2

Register reg9_i_lockdown2 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
INSTRLOCK010	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 010

Register ([L2Cpl310](#)) reg9_d_lockdown3

Name	reg9_d_lockdown3
Relative Address	0x00000918
Absolute Address	0xF8F02918
Width	32 bits

Access Type mixed
 Reset Value 0x00000000
 Description data lock down 3

Register reg9_d_lockdown3 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
DATALOCK011	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 011

Register ([L2Cpl310](#)) reg9_i_lockdown3

Name reg9_i_lockdown3
 Relative Address 0x0000091C
 Absolute Address 0xF8F0291C
 Width 32 bits
 Access Type mixed
 Reset Value 0x00000000
 Description instruction lock down 3

Register reg9_i_lockdown3 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
INSTRLOCK011	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 011

Register ([L2Cpl310](#)) reg9_d_lockdown4

Name reg9_d_lockdown4
 Relative Address 0x00000920
 Absolute Address 0xF8F02920
 Width 32 bits
 Access Type mixed
 Reset Value 0x00000000
 Description data lock down 4

Register reg9_d_lockdown4 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
DATALOCK100	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 100

Register ([L2Cpl310](#)) reg9_i_lockdown4

Name	reg9_i_lockdown4
Relative Address	0x00000924
Absolute Address	0xF8F02924
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	instruction lock down 4

Register reg9_i_lockdown4 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
INSTRLOCK100	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 100

Register ([L2Cpl310](#)) reg9_d_lockdown5

Name	reg9_d_lockdown5
Relative Address	0x00000928
Absolute Address	0xF8F02928
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	data lock down 5

Register reg9_d_lockdown5 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
DATALOCK101	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 101

Register ([L2Cpl310](#)) reg9_i_lockdown5

Name	reg9_i_lockdown5
Relative Address	0x0000092C
Absolute Address	0xF8F0292C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	instruction lock down 5

Register reg9_i_lockdown5 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
INSTRLOCK101	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 101

Register ([L2Cpl310](#)) reg9_d_lockdown6

Name	reg9_d_lockdown6
Relative Address	0x00000930
Absolute Address	0xF8F02930
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	data lock down 6

Register reg9_d_lockdown6 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
DATALOCK110	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 110

Register ([L2Cpl310](#)) reg9_i_lockdown6

Name	reg9_i_lockdown6
Relative Address	0x00000934
Absolute Address	0xF8F02934
Width	32 bits

Access Type mixed
Reset Value 0x00000000
Description instruction lock down 6

Register reg9_i_lockdown6 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
INSTRLOCK110	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 110

Register ([L2Cpl310](#)) reg9_d_lockdown7

Name reg9_d_lockdown7
Relative Address 0x00000938
Absolute Address 0xF8F02938
Width 32 bits
Access Type mixed
Reset Value 0x00000000
Description data lock down 7

Register reg9_d_lockdown7 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
DATALOCK111	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 111

Register ([L2Cpl310](#)) reg9_i_lockdown7

Name reg9_i_lockdown7
Relative Address 0x0000093C
Absolute Address 0xF8F0293C
Width 32 bits
Access Type mixed
Reset Value 0x00000000
Description instruction lock down 7

Register reg9_i_lockdown7 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
INSTRLOCK111	15:0	rw	0x0	Use when AR/WUSERSx[7:5] = 111

Register ([L2Cpl310](#)) reg9_lock_line_en

Name	reg9_lock_line_en
Relative Address	0x00000950
Absolute Address	0xF8F02950
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Lockdown by Line Enable Register.

Register reg9_lock_line_en Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	waz,r az	0x0	reserved
lock_down_by_line_en able	0	rw	0x0	0 = Lockdown by line disabled. This is the default. 1 = Lockdown by line enabled.

Register ([L2Cpl310](#)) reg9_unlock_way

Name	reg9_unlock_way
Relative Address	0x00000954
Absolute Address	0xF8F02954
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Cache lockdown by way To control the cache lockdown by way and the cache lockdown by master mechanisms see the tables from Table 3-20 to Table 3-35 on page 3-31. For these tables each bit has the following meaning: 0 allocation can occur in the corresponding way. 1 there is no allocation in the corresponding way.

Register reg9_unlock_way Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	waz,r az	0x0	reserved
unlock_all_lines_by_way_operation	15:0	rw	0x0	For all bits: 0 = Unlock all lines disabled. This is the default. 1 = Unlock all lines operation in progress for the corresponding way.

Register ([L2Cpl310](#)) reg12_addr_filtering_start

Name	reg12_addr_filtering_start
Relative Address	0x00000C00
Absolute Address	0xF8F02C00
Width	32 bits
Access Type	mixed
Reset Value	0x40000001
Description	<p>When two masters are implemented, you can redirect a whole address range to master 1 (M1).</p> <p>When address_filtering_enable is set, all accesses with address >= address_filtering_start and <address_filtering_end are automatically directed to M1. All other accesses are directed to M0.</p> <p>This feature is programmed using two registers.</p>

Register reg12_addr_filtering_start Details

Field Name	Bits	Type	Reset Value	Description
addr_filtering_start	31:20	rw	0x400	Address filtering start address.
reserved	19:1	waz,r az	0x0	reserved
addr_filtering_enable	0	rw	0x1	0 = Address filtering disabled. 1 = Address filtering enabled

Register ([L2Cpl310](#)) reg12_addr_filtering_end

Name	reg12_addr_filtering_end
Relative Address	0x00000C04
Absolute Address	0xF8F02C04
Width	32 bits
Access Type	mixed

Reset Value	0xFFF00000
Description	<p>When two masters are implemented, you can redirect a whole address range to master 1 (M1).</p> <p>When address_filtering_enable is set, all accesses with address >= address_filtering_start and <address_filtering_end are automatically directed to M1. All other accesses are directed to M0.</p> <p>This feature is programmed using two registers.</p>

Register reg12_addr_filtering_end Details

Field Name	Bits	Type	Reset Value	Description
addr_filtering_end	31:20	rw	0xFFF	Address filtering end address.
reserved	19:0	waz,r az	0x0	reserved

Register ([L2Cpl310](#)) reg15_debug_ctrl

Name	reg15_debug_ctrl
Relative Address	0x00000F40
Absolute Address	0xF8F02F40
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	<p>The Debug Control Register forces specific cache behavior required for debug. This register has</p> <p>read-only, non-secure, or read and write, secure, permission. Any secure access and non-secure</p> <p>access can read this register. Only a secure access can write to this register. If a non-secure</p> <p>access tries to write to this register the register issues a DECERR response and does not update.</p>

Register reg15_debug_ctrl Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:3	waz,r az	0x0	reserved
SPNIDEN	2	rw	0x0	Reads value of SPNIDEN input.

Field Name	Bits	Type	Reset Value	Description
DWB	1	rw	0x0	DWB: Disable write-back, force WT 0 = Enable write-back behavior. This is the default. 1 = Force write-through behavior
DCL	0	rw	0x0	DCL: Disable cache linefill 0 = Enable cache linefills. This is the default. 1 = Disable cache linefills.

Register ([L2Cpl310](#)) reg15_prefetch_ctrl

Name	reg15_prefetch_ctrl
Relative Address	0x00000F60
Absolute Address	0xF8F02F60
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	<p>Purpose Enables prefetch-related features that can improve system performance.</p> <p>Usage constraints This register has both read-only, non-secure, and read and write, secure, permissions. Any secure or non-secure access can read this register. Only a secure access can write to this register. If a non-secure access attempts to write to this register, the register</p>

Register reg15_prefetch_ctrl Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	waz,r az	0x0	reserved
double_linefill_en	30	rw	0x0	<p>Double linefill enable: You can set the following options for this register bit:</p> <p>0 The L2CC always issues 4x64-bit read bursts to L3 on reads that miss in the L2 cache. This is the default.</p> <p>1 The L2CC issues 8x64-bit read bursts to L3 on reads that miss in the L2 cache.</p>
inst_pref_en	29	rw	0x0	<p>Instruction prefetch enable: You can set the following options for this register bit:</p> <p>0 Instruction prefetching disabled. This is the default.</p> <p>1 Instruction prefetching enabled</p>

Field Name	Bits	Type	Reset Value	Description
data_pref_en	28	rw	0x0	Data prefetch enable: You can set the following options for this register bit: 0 Data prefetching disabled. This is the default. 1 Data prefetching enabled.
double_linefill_on_wrapread_en	27	rw	0x0	Double linefill on WRAP read disable: You can set the following options for this register bit: 0 Double linefill on WRAP read enabled. This is the default. 1 Double linefill on WRAP read disabled
reserved	26:25	waz,raz	0x0	reserved
pref_drop_en	24	rw	0x0	Prefetch drop enable: You can set the following options for this register bit: 0 The L2CC does not discard prefetch reads issued to L3. This is the default. 1 The L2CC discards prefetch reads issued to L3 when there is a resource conflict with explicit reads.
incr_double_linefill_en	23	rw	0x0	Incr double Linefill enable: You can set the following options for this register bit: 0 The L2CC does not issue INCR 8x64-bit read bursts to L3 on reads that miss in the L2 cache. This is the default. 1 The L2CC can issue INCR 8x64-bit read bursts to L3 on reads that miss in the L2 cache.
reserved	22	waz,raz	0x0	reserved
not_same_id_on_excl_seq_en	21	rw	0x0	Not same ID on exclusive sequence enable: You can set the following options for this register bit: 0 Read and write portions of a non-cacheable exclusive sequence have the same AXI ID when issued to L3. This is the default. 1 Read and write portions of a non-cacheable exclusive sequence do not have the same AXI ID when issued to L3.
reserved	20:5	waz,raz	0x0	reserved
prefetch_offset	4:0	rw	0x0	Default = b00000.

Register ([L2Cpl310](#)) reg15_power_ctrl

Name	reg15_power_ctrl
Relative Address	0x00000F80
Absolute Address	0xF8F02F80
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Purpose Controls the operating mode clock and power modes. Usage constraints There are no usage constraints.

Register reg15_power_ctrl Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:2	waz,r az	0x0	reserved
dynamic_clk_gating_en	1	rw	0x0	Dynamic clock gating enable. 1 = Enabled. 0 = Masked. This is the default.
standby_mode_en	0	rw	0x0	Standby mode enable. 1 = Enabled. 0 = Masked. This is the default

B.24 Application Processing Unit (mpcore)

Module Name	Application Processing Unit (mpcore)
Software Name	XSCU
Base Address	0xF8F00000 mpcore
Description	Mpcore - SCU, Interrupt controller, Counters and Timers
Version	r2p2
Doc Version	1.3
Vendor Info	ARM

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
SCU_CONTROL_REGISTER	0x00000000	32	rw	0x00000002	SCU Control Register
SCU_CONFIGURATION_REGISTER	0x00000004	32	ro	0x00000501	SCU Configuration Register
SCU_CPU_Power_Status_Register	0x00000008	32	rw	0x00000000	SCU CPU Power Status Register
SCU_Invalidate_All_Registers_in_Secure_State	0x0000000C	32	rw	0x00000000	SCU Invalidate All Registers in Secure State
Filtering_Start_Addresses_Register	0x00000040	32	rw	0x00100000	Filtering Start Address Register
Filtering_End_Address_Register	0x00000044	32	rw	0x00000000	Defined by FILTEREND input
SCU_Access_Control_Register_SAC	0x00000050	32	rw	0x0000000F	SCU Access Control (SAC) Register
SCU_Non_secure_Access_Control_Register	0x00000054	32	ro	0x00000000	SCU Non-secure Access Control Register SNSAC
ICCICR	0x00000100	32	rw	0x00000000	CPU Interface Control Register
ICCPMR	0x00000104	32	rw	0x00000000	Interrupt Priority Mask Register
ICCBPR	0x00000108	32	rw	0x00000002	Binary Point Register
ICCIAR	0x0000010C	32	rw	0x000003FF	Interrupt Acknowledge Register
ICCEOIR	0x00000110	32	rw	0x00000000	End Of Interrupt Register
ICCRPR	0x00000114	32	rw	0x000000FF	Running Priority Register

Register Name	Address	Width	Type	Reset Value	Description
ICCHPIR	0x00000118	32	rw	0x000003FF	Highest Pending Interrupt Register
ICCABPR	0x0000011C	32	rw	0x00000003	Aliased Non-secure Binary Point Register
ICCIDR	0x000001FC	32	ro	0x3901243B	CPU Interface Implementer Identification Register
Global_Timer_Counter_Register0	0x00000200	32	rw	0x00000000	Global Timer Counter Register 0
Global_Timer_Counter_Register1	0x00000204	32	rw	0x00000000	Global Timer Counter Register 1
Global_Timer_Control_Register	0x00000208	32	rw	0x00000000	Global Timer Control Register
Global_Timer_Interrupt_Status_Register	0x0000020C	32	rw	0x00000000	Global Timer Interrupt Status Register
Comparator_Value_Register0	0x00000210	32	rw	0x00000000	Comparator Value Register_0
Comparator_Value_Register1	0x00000214	32	rw	0x00000000	Comparator Value Register_1
Auto_increment_Register	0x00000218	32	rw	0x00000000	Auto-increment Register
Private_Timer_Load_Register	0x00000600	32	rw	0x00000000	Private Timer Load Register
Private_Timer_Counter_Register	0x00000604	32	rw	0x00000000	Private Timer Counter Register
Private_Timer_Control_Register	0x00000608	32	rw	0x00000000	Private Timer Control Register
Private_Timer_Interrupt_Status_Register	0x0000060C	32	rw	0x00000000	Private Timer Interrupt Status Register
Watchdog_Load_Register	0x00000620	32	rw	0x00000000	Watchdog Load Register
Watchdog_Counter_Register	0x00000624	32	rw	0x00000000	Watchdog Counter Register
Watchdog_Control_Register	0x00000628	32	rw	0x00000000	Watchdog Control Register
Watchdog_Interrupt_Status_Register	0x0000062C	32	rw	0x00000000	Watchdog Interrupt Status Register
Watchdog_Reset_Status_Register	0x00000630	32	rw	0x00000000	Watchdog Reset Status Register
Watchdog_Disable_Register	0x00000634	32	rw	0x00000000	Watchdog Disable Register
ICDDCR	0x00001000	32	rw	0x00000000	Distributor Control Register

Register Name	Address	Width	Type	Reset Value	Description
ICDICTR	0x00001004	32	ro	0x00000C22	Interrupt Controller Type Register
ICDIIDR	0x00001008	32	ro	0x0102043B	Distributor Implementer Identification Register
ICDISR0	0x00001080	32	rw	0x00000000	Interrupt Security Register_0
ICDISR1	0x00001084	32	rw	0x00000000	Interrupt Security Register_1
ICDISR2	0x00001088	32	rw	0x00000000	Interrupt Security Register_2
ICDISER0	0x00001100	32	rw	0x0000FFFF	Interrupt Set-enable Register 0
ICDISER1	0x00001104	32	rw	0x00000000	Interrupt Set-enable Register 1
ICDISER2	0x00001108	32	rw	0x00000000	Interrupt Set-enable Register 2
ICDICER0	0x00001180	32	rw	0x0000FFFF	Interrupt Clear-Enable Register 0
ICDICER1	0x00001184	32	rw	0x00000000	Interrupt Clear-Enable Register 1
ICDICER2	0x00001188	32	rw	0x00000000	Interrupt Clear-Enable Register 2
ICDISPR0	0x00001200	32	rw	0x00000000	Interrupt Set-pending Register_0
ICDISPR1	0x00001204	32	rw	0x00000000	Interrupt Set-pending Register_1
ICDISPR2	0x00001208	32	rw	0x00000000	Interrupt Set-pending Register_2
ICDICPR0	0x00001280	32	rw	0x00000000	Interrupt Clear-Pending Register_0
ICDICPR1	0x00001284	32	rw	0x00000000	Interrupt Clear-Pending Register_1
ICDICPR2	0x00001288	32	rw	0x00000000	Interrupt Clear-Pending Register_2
ICDABR0	0x00001300	32	rw	0x00000000	Active Bit register_0
ICDABR1	0x00001304	32	rw	0x00000000	Active Bit register_1
ICDABR2	0x00001308	32	rw	0x00000000	Active Bit register_2
ICDIPR0	0x00001400	32	rw	0x00000000	Interrupt Priority Register_0
ICDIPR1	0x00001404	32	rw	0x00000000	Interrupt Priority Register_1
ICDIPR2	0x00001408	32	rw	0x00000000	Interrupt Priority Register_2
ICDIPR3	0x0000140C	32	rw	0x00000000	Interrupt Priority Register_3
ICDIPR4	0x00001410	32	rw	0x00000000	Interrupt Priority Register_4
ICDIPR5	0x00001414	32	rw	0x00000000	Interrupt Priority Register_5
ICDIPR6	0x00001418	32	rw	0x00000000	Interrupt Priority Register_6

Register Name	Address	Width	Type	Reset Value	Description
ICDIPR7	0x0000141C	32	rw	0x00000000	Interrupt Priority Register_7
ICDIPR8	0x00001420	32	rw	0x00000000	Interrupt Priority Register_8
ICDIPR9	0x00001424	32	rw	0x00000000	Interrupt Priority Register_9
ICDIPR10	0x00001428	32	rw	0x00000000	Interrupt Priority Register_10
ICDIPR11	0x0000142C	32	rw	0x00000000	Interrupt Priority Register_11
ICDIPR12	0x00001430	32	rw	0x00000000	Interrupt Priority Register_12
ICDIPR13	0x00001434	32	rw	0x00000000	Interrupt Priority Register_13
ICDIPR14	0x00001438	32	rw	0x00000000	Interrupt Priority Register_14
ICDIPR15	0x0000143C	32	rw	0x00000000	Interrupt Priority Register_15
ICDIPR16	0x00001440	32	rw	0x00000000	Interrupt Priority Register_16
ICDIPR17	0x00001444	32	rw	0x00000000	Interrupt Priority Register_17
ICDIPR18	0x00001448	32	rw	0x00000000	Interrupt Priority Register_18
ICDIPR19	0x0000144C	32	rw	0x00000000	Interrupt Priority Register_19
ICDIPR20	0x00001450	32	rw	0x00000000	Interrupt Priority Register_20
ICDIPR21	0x00001454	32	rw	0x00000000	Interrupt Priority Register_21
ICDIPR22	0x00001458	32	rw	0x00000000	Interrupt Priority Register_22
ICDIPR23	0x0000145C	32	rw	0x00000000	Interrupt Priority Register_23
ICDIPTR0	0x00001800	32	rw	0x01010101	Interrupt Processor Targets Register_0
ICDIPTR1	0x00001804	32	rw	0x01010101	Interrupt Processor Targets Register_1
ICDIPTR2	0x00001808	32	rw	0x01010101	Interrupt Processor Targets Register_2
ICDIPTR3	0x0000180C	32	rw	0x01010101	Interrupt Processor Targets Register_3
ICDIPTR4	0x00001810	32	rw	0x01010101	Interrupt Processor Targets Register_4
ICDIPTR5	0x00001814	32	rw	0x01010101	Interrupt Processor Targets Register_5
ICDIPTR6	0x00001818	32	rw	0x01010101	Interrupt Processor Targets Register_6
ICDIPTR7	0x0000181C	32	rw	0x01010101	Interrupt Processor Targets Register_7
ICDIPTR8	0x00001820	32	rw	0x01010101	Interrupt Processor Targets Register_8
ICDIPTR9	0x00001824	32	rw	0x01010101	Interrupt Processor Targets Register_9

Register Name	Address	Width	Type	Reset Value	Description
ICDIPTTR10	0x00001828	32	rw	0x01010101	Interrupt Processor Targets Register_10
ICDIPTTR11	0x0000182C	32	rw	0x01010101	Interrupt Processor Targets Register_11
ICDIPTTR12	0x00001830	32	rw	0x01010101	Interrupt Processor Targets Register_12
ICDIPTTR13	0x00001834	32	rw	0x01010101	Interrupt Processor Targets Register_13
ICDIPTTR14	0x00001838	32	rw	0x01010101	Interrupt Processor Targets Register_14
ICDIPTTR15	0x0000183C	32	rw	0x01010101	Interrupt Processor Targets Register_15
ICDIPTTR16	0x00001840	32	rw	0x01010101	Interrupt Processor Targets Register_16
ICDIPTTR17	0x00001844	32	rw	0x01010101	Interrupt Processor Targets Register_17
ICDIPTTR18	0x00001848	32	rw	0x01010101	Interrupt Processor Targets Register_18
ICDIPTTR19	0x0000184C	32	rw	0x01010101	Interrupt Processor Targets Register_19
ICDIPTTR20	0x00001850	32	rw	0x01010101	Interrupt Processor Targets Register_20
ICDIPTTR21	0x00001854	32	rw	0x01010101	Interrupt Processor Targets Register_21
ICDIPTTR22	0x00001858	32	rw	0x01010101	Interrupt Processor Targets Register_22
ICDIPTTR23	0x0000185C	32	rw	0x01010101	Interrupt Processor Targets Register_23
ICDICFR0	0x00001C00	32	rw	0xAAAAAAAAA	Interrupt Configuration Register 0
ICDICFR1	0x00001C04	32	rw	0x7DC00000	Interrupt Configuration Register 1
ICDICFR2	0x00001C08	32	rw	0x55555555	Interrupt Configuration Register 2
ICDICFR3	0x00001C0C	32	rw	0x55555555	Interrupt Configuration Register 3
ICDICFR4	0x00001C10	32	rw	0x55555555	Interrupt Configuration Register 4
ICDICFR5	0x00001C14	32	rw	0x55555555	Interrupt Configuration Register 5
ppi_status	0x00001D00	32	ro	0x00000000	PPI Status Register

Register Name	Address	Width	Type	Reset Value	Description
spi_status_0	0x00001D04	32	ro	0x00000000	SPI Status Register 0
spi_status_1	0x00001D08	32	ro	0x00000000	SPI Status Register 1
ICDSGIR	0x00001F00	32	rw	0x00000000	Software Generated Interrupt Register
ICPIDR4	0x00001FD0	32	rw	0x00000004	Peripheral ID4
ICPIDR5	0x00001FD4	32	rw	0x00000000	Peripheral ID5
ICPIDR6	0x00001FD8	32	rw	0x00000000	Peripheral ID6
ICPIDR7	0x00001FDC	32	rw	0x00000000	Peripheral ID7
ICPIDR0	0x00001FE0	32	rw	0x00000090	Peripheral ID0
ICPIDR1	0x00001FE4	32	rw	0x000000B3	Peripheral ID1
ICPIDR2	0x00001FE8	32	rw	0x0000001B	Peripheral ID2
ICPIDR3	0x00001FEC	32	rw	0x00000000	Peripheral ID3
ICCIDR0	0x00001FF0	32	rw	0x0000000D	Component ID0
ICCIDR1	0x00001FF4	32	rw	0x000000F0	Component ID1
ICCIDR2	0x00001FF8	32	rw	0x00000005	Component ID2
ICCIDR3	0x00001FFC	32	rw	0x000000B1	Component ID3

Register ([mpcore](#)) SCU_CONTROL_REGISTER

Name	SCU_CONTROL_REGISTER
Relative Address	0x00000000
Absolute Address	0xF8F00000
Width	32 bits
Access Type	rw
Reset Value	0x00000002
Description	SCU Control Register

Register SCU_CONTROL_REGISTER Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:7	rw	0x0	Reserved. Writes are ignored, read data is always zero.
IC_standby_enable	6	rw	0x0	When set, this stops the Interrupt Controller clock when no interrupts are pending, and no CPU is performing a read/write request.

Field Name	Bits	Type	Reset Value	Description
SCU_standby_enable	5	rw	0x0	When set, SCU CLK is turned off when all processors are in WFI mode, there is no pending request on the ACP (if implemented), and there is no remaining activity in the SCU. When SCU CLK is off, ARREADYs, AWREADYs and WREADYs on the ACP are forced LOW. The clock is turned on when any processor leaves WFI mode, or if there is a new request on the ACP.
Force_all_Device_to_port0_enable	4	rw	0x0	When set, all requests from the ACP or processors with AxCACHE = NonCacheable Bufferable are forced to be issued on the AXI Master port M0.
SCU_Speculative_linefills_enable	3	rw	0x0	When set, coherent linefill requests are sent speculatively to the L2C-310 in parallel with the tag look-up. If the tag look-up misses, the confirmed linefill is sent to the L2C-310 and gets RDATA earlier because the data request was already initiated by the speculative request. This feature works only if the L2C-310 is present in the design.
SCU_RAMs_Parity_enable	2	rw	0x0	1 = Parity on. 0 = Parity off. This is the default setting. This bit is always zero if support for parity is not implemented.
Address_filtering_enable	1	rw	0x1	1 = Addressing filtering on. 0 = Addressing filtering off. The default value is the value of FILTEREN sampled when nSCURESET is deasserted. This bit is always zero if the SCU is implemented in the single master port configuration.
SCU_enable	0	rw	0x0	1 = SCU enable. 0 = SCU disable. This is the default setting

Register ([mpcore](#)) SCU_CONFIGURATION_REGISTER

Name SCU_CONFIGURATION_REGISTER

Relative Address	0x00000004
Absolute Address	0xF8F00004
Width	32 bits
Access Type	ro
Reset Value	0x00000501
Description	SCU Configuration Register

Register SCU_CONFIGURATION_REGISTER Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Should Be Zero (SBZ)
Tag_RAM_sizes	15:8	ro	0x5	<p>Bits [15:14] indicate Cortex-A9 processor CPU3 tag RAM size if present.</p> <p>Bits [13:12] indicate Cortex-A9 processor CPU2 tag RAM size if present.</p> <p>Bits [11:10] indicate Cortex-A9 processor CPU1 tag RAM size if present.</p> <p>Bits [9:8] indicate Cortex-A9 processor CPU0 tag RAM size.</p> <p>The encoding is as follows:</p> <p>b11 = reserved</p> <p>b10 = 64KB cache, 256 indexes per tag RAM</p> <p>b01 = 32KB cache, 128 indexes per tag RAM</p> <p>b00 = 16KB cache, 64 indexes per tag RAM.</p>
CPUs_SMP	7:4	ro	0x0	<p>Shows the Cortex-A9 processors that are in Symmetric Multi-processing (SMP) or Asymmetric Multi-processing (AMP) mode.</p> <p>0 = this Cortex-A9 processor is in AMP mode not taking part in coherency or not present.</p> <p>1 = this Cortex-A9 processor is in SMP mode taking part in coherency.</p> <p>Bit 7 is for CPU3</p> <p>Bit 6 is for CPU2</p> <p>Bit 5 is for CPU1</p> <p>Bit 4 is for CPU0.</p>
reserved	3:2	ro	0x0	Should Be Zero (SBZ)
CPU_number	1:0	ro	0x1	<p>Number of CPUs present in the Cortex-A9 MPCore processor</p> <p>b11 = four Cortex-A9 processors, CPU0, CPU1, CPU2, and CPU3</p> <p>b10 = three Cortex-A9 processors, CPU0, CPU1, and CPU2</p> <p>b01 = two Cortex-A9 processors, CPU0 and CPU1</p> <p>b00 = one Cortex-A9 processor, CPU0.</p>

Register ([mpcore](#)) SCU_CPU_Power_Status_Register

Name	SCU_CPU_Power_Status_Register
Relative Address	0x00000008
Absolute Address	0xF8F00008
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	SCU CPU Power Status Register

Register SCU_CPU_Power_Status_Register Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	rw	0x0	Should Be Zero (SBZ)
CPU3_status	25:24	rw	0x0	Power status of the Cortex-A9 processor: b00: Normal mode. b01: Reserved. b10: the Cortex-A9 processor is about to enter (or is in) dormant mode. No coherency request is sent to the Cortex-A9 processor. b11: the Cortex-A9 processor is about to enter (or is in) powered-off mode, or is nonpresent. No coherency request is sent to the Cortex-A9 processor.
reserved	23:18	rw	0x0	Should Be Zero (SBZ)
CPU2_status	17:16	rw	0x0	Power status of the Cortex-A9 processor
reserved	15:10	rw	0x0	Should Be Zero (SBZ)
CPU1_status	9:8	rw	0x0	Power status of the Cortex-A9 processor
reserved	7:2	rw	0x0	Should Be Zero (SBZ)
CPU0_status	1:0	rw	0x0	Power status of the Cortex-A9 processor

Register ([mpcore](#)) SCU_Invalidate_All_Registers_in_Secure_State

Name	SCU_Invalidate_All_Registers_in_Secure_State
Relative Address	0x0000000C
Absolute Address	0xF8F0000C
Width	32 bits
Access Type	rw
Reset Value	0x00000000

Description SCU Invalidate All Registers in Secure State

Register SCU_Invalidate_All_Registers_in_Secure_State Details

Field Name	Bits	Type	Reset Value	Description
NA	31:16	rw	0x0	NA
CPU3_ways	15:12	rw	0x0	Specifies the ways that must be invalidated for CPU3. Writing to these bits has no effect if the Cortex-A9 MPCore processor has fewer than four processors.
CPU2_ways	11:8	rw	0x0	Specifies the ways that must be invalidated for CPU2. Writing to these bits has no effect if the Cortex-A9 MPCore processor has fewer than three processors
CPU1_ways	7:4	rw	0x0	Specifies the ways that must be invalidated for CPU1. Writing to these bits has no effect if the Cortex-A9 MPCore processor has fewer than two processors.
CPU0_ways	3:0	rw	0x0	Specifies the ways that must be invalidated for CPU0

Register ([mpcore](#)) Filtering_Start_Address_Register

Name Filtering_Start_Address_Register
Relative Address 0x00000040
Absolute Address 0xF8F00040
Width 32 bits
Access Type rw
Reset Value 0x00100000
Description Filtering Start Address Register

Register Filtering_Start_Address_Register Details

Field Name	Bits	Type	Reset Value	Description
Filtering_start_address	31:20	rw	0x1	Start address for use with master port 1 in a two-master port configuration when address filtering is enabled. The default value is the value of FILTERSTART sampled on exit from reset. The value on the pin gives the upper address bits with 1MB granularity.
SBZ	19:0	rw	0x0	SBZ

Register ([mpcore](#)) Filtering_End_Address_Register

Name	Filtering_End_Address_Register
Relative Address	0x00000044
Absolute Address	0xF8F00044
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Defined by FILTEREND input

Register Filtering_End_Address_Register Details

Field Name	Bits	Type	Reset Value	Description
Filtering_end_address	31:20	rw	0x0	End address for use with master port 1 in a two-master port configuration, when address filtering is enabled. The default value is the value of FILTEREND sampled on exit from reset. The value on the pin gives the upper address bits with 1MB granularity.
SBZ	19:0	rw	0x0	SBZ

Register ([mpcore](#)) SCU_Access_Control_Register_SAC

Name	SCU_Access_Control_Register_SAC
Relative Address	0x00000050
Absolute Address	0xF8F00050
Width	32 bits
Access Type	rw
Reset Value	0x0000000F
Description	SCU Access Control (SAC) Register

Register SCU_Access_Control_Register_SAC Details

Field Name	Bits	Type	Reset Value	Description
	31:4	rw	0x0	SBZ
CPU3	3	rw	0x1	0 = CPU3 cannot access the components. 1 = CPU3 can access the components. This is the default.

Field Name	Bits	Type	Reset Value	Description
CPU2	2	rw	0x1	0 = CPU2 cannot access the components. 1 = CPU2 can access the components. This is the default.
CPU1	1	rw	0x1	0 = CPU1 cannot access the components. 1 = CPU1 can access the components. This is the default.
CPU0	0	rw	0x1	0 = CPU0 cannot access the components. 1 = CPU0 can access the components. This is the default.

Register ([mpcore](#)) SCU_Non_secure_Access_Control_Register

Name	SCU_Non_secure_Access_Control_Register
Relative Address	0x00000054
Absolute Address	0xF8F00054
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	SCU Non-secure Access Control Register SNSAC

Register SCU_Non_secure_Access_Control_Register Details

Field Name	Bits	Type	Reset Value	Description
SBZ	31:12	ro	0x0	SBZ
CPU3_global_timer	11	ro	0x0	same as above
CPU2_global_timer	10	ro	0x0	same as above
CPU1_global_timer	9	ro	0x0	same as above
CPU0_global_timer	8	ro	0x0	Non-secure access to the global timer for CPU<n>. * <n> is 3 for bit[11] * <n> is 2 for bit[10] * <n> is 1 for bit[9] * <n> is 0 for bit[8]. 0 = Secure accesses only. This is the default value. 1 = Secure accesses and Non-Secure accesses.
Private_timers_for_CPU3	7	ro	0x0	same as above
Private_timers_for_CPU2	6	ro	0x0	same as above

Field Name	Bits	Type	Reset Value	Description
Private_timers_for_CP U1	5	ro	0x0	same as above
Private_timers_for_CP U0	4	ro	0x0	Non-secure access to the private timer and watchdog for CPU<n>. * <n> is 3 for bit[7] * <n> is 2 for bit[6] * <n> is 1 for bit[5] * <n> is 0 for bit[4]. 0 = Secure accesses only. Non-secure reads return 0. This is the default value. 1 = Secure accesses and Non-secure accesses.
Component_access_for _CPU3	3	ro	0x0	same as above
Component_access_for _CPU2	2	ro	0x0	same as above
Component_access_for _CPU1	1	ro	0x0	same as above
Component_access_for _CPU0	0	ro	0x0	Non-secure access to the components for CPU<n>. * <n> is 3 for bit[3] * <n> is 2 for bit[2] * <n> is 1 for bit[1] * <n> is 0 for bit[0]. 0 = CPU cannot write the components 1 = CPU can access the components.

Register ([mpcore](#)) ICCICR

Name	ICCICR
Software Name	GIC_CONTROL
Relative Address	0x00000100
Absolute Address	0xF8F00100
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	CPU Interface Control Register

Register ICCICR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:5	rw	0x0	reserved
SBPR (GIC_CNTR_SBPR)	4	rw	0x0	Controls whether the CPU interface uses the Secure or Non-secure Binary Point Register for preemption. 0: use the Secure Binary Point Register for Secure interrupts, and use the Non-secure Binary Point Register for Non-secure interrupts. 1: use the Secure Binary Point Register for both Secure and Non-secure interrupts.
FIQEn (GIC_CNTR_FIQEN)	3	rw	0x0	Controls whether the GIC signals Secure interrupts to a target processor using the FIQ or the IRQ signal. 0: using IRQ, 1: using FIQ. The GIC always signals Non-secure interrupts using IRQ.
AckCtl (GIC_CNTR_ACKCTL)	2	rw	0x0	Controls whether a Secure read of the ICCIAR, when the highest priority pending interrupt is Non-secure, causes the CPU interface to acknowledge the interrupt.
EnableNS (GIC_CNTR_EN_NS)	1	rw	0x0	An alias of the Enable bit in the Non-secure ICCICR. This alias bit means Secure software can enable the signal of Non-secure interrupts.
EnableS (GIC_CNTR_EN_S)	0	rw	0x0	Global enable for the signaling of Secure interrupts by the CPU interfaces to the connected processors.

Register ([mpcore](#)) ICCPMR

Name	ICCPMR
Software Name	GIC_CPU_PRIOR
Relative Address	0x00000104
Absolute Address	0xF8F00104
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Interrupt Priority Mask Register

Register ICCPMR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rw	0x0	reserved
Priority (GIC_PRIORITY)	7:0	rw	0x0	The priority mask level for the CPU interface. If the priority of an interrupt is higher than the value indicated by this field, the interface signals the interrupt to the processor.

Register ([mpcore](#)) ICCBPR

Name	ICCBPR
Software Name	GIC_BIN_PT
Relative Address	0x00000108
Absolute Address	0xF8F00108
Width	32 bits
Access Type	rw
Reset Value	0x00000002
Description	Binary Point Register

Register ICCBPR Details

Field Name	Bits	Type	Reset Value	Description
reserved (GIC_CPUID)	31:3	rw	0x0	reserved
Binary_point (MASK)	2:0	rw	0x2	The value of this field controls the 8-bit interrupt priority field is split into a group priority field, used to determine interrupt preemption, and a subpriority field.

Register ([mpcore](#)) ICCIAR

Name	ICCIAR
Software Name	GIC_INT_ACK
Relative Address	0x0000010C
Absolute Address	0xF8F0010C
Width	32 bits
Access Type	rw
Reset Value	0x000003FF
Description	Interrupt Acknowledge Register

Register ICCIAR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:13	rw	0x0	reserved
CPUID	12:10	rw	0x0	Identifies the processor that requested the interrupt. Returns the number of the CPU interface that made the request.
ACKINTID (GIC_ACK_INTID)	9:0	rw	0x3FF	The interrupt ID. This read acts as an acknowledge for the interrupt.

Register ([mpcore](#)) ICCEOIR

Name	ICCEOIR
Software Name	GIC_EOI
Relative Address	0x00000110
Absolute Address	0xF8F00110
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	End Of Interrupt Register

Register ICCEOIR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:13	rw	0x0	reserved
CPUID	12:10	rw	0x0	On completion of the processing of an SGI, this field contains the CPUID value from the corresponding ICCIAR access.
EOIINTID (INTID)	9:0	rw	0x0	The ACKINTID value from the corresponding ICCIAR access.

Register ([mpcore](#)) ICCRPR

Name	ICCRPR
Software Name	GIC_RUN_PRIOR
Relative Address	0x00000114
Absolute Address	0xF8F00114
Width	32 bits

Access Type rw
Reset Value 0x000000FF
Description Running Priority Register

Register ICCRPR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rw	0x0	reserved
Priority (GIC_RUN_PRIORITY)	7:0	rw	0xFF	The priority value of the highest priority interrupt that is active on the CPU interface.

Register ([mpcore](#)) ICCHPIR

Name ICCHPIR
Software Name GIC_HI_PEND
Relative Address 0x00000118
Absolute Address 0xF8F00118
Width 32 bits
Access Type rw
Reset Value 0x000003FF
Description Highest Pending Interrupt Register

Register ICCHPIR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:13	rw	0x0	reserved
CPUID (GIC_CPUID)	12:10	rw	0x0	If the PENDINTID field returns the ID of an SGI, this field contains the CPUID value for that interrupt. The identifies the processor that generated the interrupt.
PENDINTID (GIC_PEND_INTID)	9:0	rw	0x3FF	The interrupt ID of the highest priority pending interrupt.

Register ([mpcore](#)) ICCABPR

Name ICCABPR
Software Name GIC_ALIAS_BIN_PT
Relative Address 0x0000011C

Absolute Address	0xF8F0011C
Width	32 bits
Access Type	rw
Reset Value	0x00000003
Description	Aliased Non-secure Binary Point Register

Register ICCABPR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:3	rw	0x0	reserved
Binary_point	2:0	rw	0x3	Provides an alias of the Non-secure ICCBPR.

Register ([mpcore](#)) ICCIDR

Name	ICCIDR
Relative Address	0x000001FC
Absolute Address	0xF8F001FC
Width	32 bits
Access Type	ro
Reset Value	0x3901243B
Description	CPU Interface Implementer Identification Register

Register ICCIDR Details

Field Name	Bits	Type	Reset Value	Description
Part_number	31:20	ro	0x390	Identifies the peripheral
Architecture_version	19:16	ro	0x1	Identifies the architecture version
Revision_number	15:12	ro	0x2	Returns the revision number of the Interrupt Controller. The implementer defines the format of this field.
Implementer	11:0	ro	0x43B	Returns the JEP106 code of the company that implemented the Cortex-A9 processor interface RTL. It uses the following construct: [11:8] the JEP106 continuation code of the implementer [7] 0 [6:0] the JEP106 code [6:0] of the implementer

Register ([mpcore](#)) Global_Timer_Counter_Register0

Name	Global_Timer_Counter_Register0
Relative Address	0x00000200
Absolute Address	0xF8F00200
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Global Timer Counter Register 0

Note: This register is the first in an array of 2 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
Global_Timer_Counter_Register0	0xf8f00200
Global_Timer_Counter_Register1	0xf8f00204

Register Global_Timer_Counter_Register0 to Global_Timer_Counter_Register1 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>There are two timer counter registers. They are the lower 32-bit timer counter at offset 0x00 and the upper 32-bit timer counter at offset 0x04.</p> <p>You must access these registers with 32-bit accesses. You cannot use STRD/LDRD.</p> <p>To modify the register proceed as follows:</p> <ol style="list-style-type: none"> 1. Clear the timer enable bit in the Global Timer Control Register 2. Write the lower 32-bit timer counter register 3. Write the upper 32-bit timer counter register 4. Set the timer enable bit. <p>To get the value from the Global Timer Counter register proceed as follows:</p> <ol style="list-style-type: none"> 1. Read the upper 32-bit timer counter register 2. Read the lower 32-bit timer counter register 3. Read the upper 32-bit timer counter register again. If the value is different to the 32-bit upper value read previously, go back to step 2. Otherwise the 64-bit timer counter value is correct.

Register ([mpcore](#)) Global_Timer_Control_Register

Name	Global_Timer_Control_Register
Relative Address	0x00000208
Absolute Address	0xF8F00208
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Global Timer Control Register

Register Global_Timer_Control_Register Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	rw	0x0	Reserved
Prescaler	15:8	rw	0x0	The prescaler modifies the clock period for the decrementing event for the Counter Register. See Calculating timer intervals on page 4-2 for the equation
reserved	7:4	rw	0x0	Reserved
	3	rw	0x0	This bit is banked per Cortex-A9 processor. 1'b0: single shot mode. When the counter reaches the comparator value, sets the event flag. It is the responsibility of software to update the comparator value to get further events. 1'b1: auto increment mode. Each time the counter reaches the comparator value, the comparator register is incremented with the auto-increment register, so that further events can be set periodically without any software updates.
IRQ_Enable	2	rw	0x0	This bit is banked per Cortex-A9 processor. If set, the interrupt ID 27 is set as pending in the Interrupt Distributor when the event flag is set in the Timer Status Register.

Field Name	Bits	Type	Reset Value	Description
Comp_Enablea	1	rw	0x0	This bit is banked per Cortex-A9 processor. If set, it allows the comparison between the 64-bit Timer Counter and the related 64-bit Comparator Register.
Timer_Enable	0	rw	0x0	Timer enable 1'b0 = Timer is disabled and the counter does not increment. All registers can still be read and written 1'b1 = Timer is enabled and the counter increments normally

Register ([mpcore](#)) Global_Timer_Interrupt_Status_Register

Name	Global_Timer_Interrupt_Status_Register
Relative Address	0x0000020C
Absolute Address	0xF8F0020C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Global Timer Interrupt Status Register

Register Global_Timer_Interrupt_Status_Register Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	UNK/SBZP
Event_flag	0	rw	0x0	This is a banked register for all Cortex-A9 processors present. The event flag is a sticky bit that is automatically set when the Counter Register reaches the Comparator Register value. If the timer interrupt is enabled, Interrupt ID 27 is set as pending in the Interrupt Distributor after the event flag is set. The event flag is cleared when written to 1. Figure 4-7 shows the Global Timer Interrupt Status Register bit assignment.

Register ([mpcore](#)) Comparator_Value_Register0

Name	Comparator_Value_Register0
Relative Address	0x00000210

Absolute Address	0xF8F00210
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Comparator Value Register_0

Note: This register is the first in an array of 2 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
Comparator_Value_Register0	0xf8f00210
Comparator_Value_Register1	0xf8f00214

Register Comparator_Value_Register0 to Comparator_Value_Register1 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>There are two 32-bit registers, the lower 32-bit comparator value register at offset 0x10 and the upper 32-bit comparator value register at offset 0x14.</p> <p>You must access these registers with 32-bit accesses. You cannot use STRD/LDRD. There is a Comparator Value Register for each Cortex-A9 processor.</p> <p>To ensure that updates to this register do not set the Interrupt Status Register proceed as follows:</p> <ol style="list-style-type: none"> 1. Clear the Comp Enable bit in the Timer Control Register. 2. Write the lower 32-bit Comparator Value Register. 3. Write the upper 32-bit Comparator Value Register. 4. Set the Comp Enable bit and, if necessary, the IRQ enable bit.

Register ([mpcore](#)) Auto_increment_Register

Name	Auto_increment_Register
Relative Address	0x00000218
Absolute Address	0xF8F00218
Width	32 bits

Access Type rw
Reset Value 0x00000000
Description Auto-increment Register

Register Auto_increment_Register Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Auto-increment Register This 32-bit register gives the increment value of the Comparator Register when the Auto-increment bit is set in the Timer Control Register. Each Cortex-A9 processor present has its own Auto-increment Register. If the comp enable and auto-increment bits are set when the global counter reaches the Comparator Register value, the comparator is incremented by the auto-increment value, so that a new event can be set periodically. The global timer is not affected and goes on incrementing

Register ([mpcore](#)) Private_Timer_Load_Register

Name Private_Timer_Load_Register
Software Name TIMER_LOAD
Relative Address 0x00000600
Absolute Address 0xF8F00600
Width 32 bits
Access Type rw
Reset Value 0x00000000
Description Private Timer Load Register

Register Private_Timer_Load_Register Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	The Timer Load Register contains the value copied to the Timer Counter Register when it decrements down to zero with auto reload mode enabled. Writing to the Timer Load Register means that you also write to the Timer Counter Register.

Register ([mpcore](#)) Private_Timer_Counter_Register

Name	Private_Timer_Counter_Register
Software Name	TIMER_COUNTER
Relative Address	0x00000604
Absolute Address	0xF8F00604
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Private Timer Counter Register

Register Private_Timer_Counter_Register Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>The Timer Counter Register is a decrementing counter.</p> <p>The Timer Counter Register decrements if the timer is enabled using the timer enable bit in the Timer Control Register. If a Cortex-A9 processor timer is in debug state, the counter only decrements when the Cortex-A9 processor returns to non debug state.</p> <p>When the Timer Counter Register reaches zero and auto reload mode is enabled, it reloads the value in the Timer Load Register and then decrements from that value. If auto reload mode is not enabled, the Timer Counter Register decrements down to zero and stops.</p> <p>When the Timer Counter Register reaches zero, the timer interrupt status event flag is set and the interrupt ID 29 is set as pending in the Interrupt Distributor, if interrupt generation is enabled in the Timer Control Register.</p> <p>Writing to the Timer Counter Register or Timer Load Register forces the Timer Counter Register to decrement from the newly written value.</p>

Register ([mpcore](#)) Private_Timer_Control_Register

Name	Private_Timer_Control_Register
Software Name	TIMER_CONTROL

Relative Address	0x00000608
Absolute Address	0xF8F00608
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Private Timer Control Register

Register Private_Timer_Control_Register Details

Field Name	Bits	Type	Reset Value	Description
SBZP	31:16	rw	0x0	UNK/SBZP.
Prescaler (PRESCALER)	15:8	rw	0x0	The prescaler modifies the clock period for the decrementing event for the Counter Register. See Calculating timer intervals on page 4-2 for the equation.\
UNK_SBZP	7:3	rw	0x0	UNK/SBZP.
IRQ_Enable (IRQ_ENABLE)	2	rw	0x0	If set, the interrupt ID 29 is set as pending in the Interrupt Distributor when the event flag is set in the Timer Status Register.
Auto_reload (AUTO_RELOAD)	1	rw	0x0	1'b0 = Single shot mode. Counter decrements down to zero, sets the event flag and stops. 1'b1 = Auto-reload mode. Each time the Counter Register reaches zero, it is reloaded with the value contained in the Timer Load Register.
Timer_Enable (ENABLE)	0	rw	0x0	Timer enable 1'b0 = Timer is disabled and the counter does not decrement. All registers can still be read and written 1'b1 = Timer is enabled and the counter decrements normally

Register ([mpcore](#)) Private_Timer_Interrupt_Status_Register

Name	Private_Timer_Interrupt_Status_Register
Software Name	TIMER_ISR
Relative Address	0x0000060C
Absolute Address	0xF8F0060C
Width	32 bits
Access Type	rw

Reset Value 0x00000000

Description Private Timer Interrupt Status Register

Register Private_Timer_Interrupt_Status_Register Details

Field Name	Bits	Type	Reset Value	Description
UNK_SBZP	31:1	rw	0x0	UNK/SBZP
	0	rw	0x0	This is a banked register for all Cortex-A9 processors present. The event flag is a sticky bit that is automatically set when the Counter Register reaches zero. If the timer interrupt is enabled, Interrupt ID 29 is set as pending in the Interrupt Distributor after the event flag is set. The event flag is cleared when written to 1.

Register ([mpcore](#)) Watchdog_Load_Register

Name Watchdog_Load_Register

Software Name WDT_LOAD

Relative Address 0x00000620

Absolute Address 0xF8F00620

Width 32 bits

Access Type rw

Reset Value 0x00000000

Description Watchdog Load Register

Register Watchdog_Load_Register Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	Watchdog Load Register The Watchdog Load Register contains the value copied to the Watchdog Counter Register when it decrements down to zero with auto reload mode enabled, in Timer mode. Writing to the Watchdog Load Register means that you also write to the Watchdog Counter Register

Register ([mpcore](#)) Watchdog_Counter_Register

Name Watchdog_Counter_Register

Software Name	WDT_COUNTER
Relative Address	0x00000624
Absolute Address	0xF8F00624
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Watchdog Counter Register

Register Watchdog_Counter_Register Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Watchdog Counter Register</p> <p>The Watchdog Counter Register is a down counter.</p> <p>It decrements if the Watchdog is enabled using the Watchdog enable bit in the Watchdog Control Register. If the Cortex-A9 processor associated with the Watchdog is in debug state, the counter does not decrement until the Cortex-A9 processor returns to non debug state.</p> <p>When the Watchdog Counter Register reaches zero and auto reload mode is enabled, and in timer mode, it reloads the value in the Watchdog Load Register and then decrements from that value. If auto reload mode is not enabled or the watchdog is not in timer mode, the Watchdog Counter Register decrements down to zero and stops.</p> <p>When in watchdog mode the only way to update the Watchdog Counter Register is to write to the Watchdog Load Register. When in timer mode the Watchdog Counter Register is write accessible.</p> <p>The behavior of the watchdog when the Watchdog Counter Register reaches zero depends on its current mode:</p> <p>Timer mode When the Watchdog Counter Register reaches zero, the watchdog interrupt status event flag is set and the interrupt ID 30 is set as pending in the Interrupt Distributor, if interrupt generation is enabled in the Watchdog Control Register.</p> <p>Watchdog mode</p> <p>If a software failure prevents the Watchdog Counter Register from being refreshed, the Watchdog Counter Register reaches zero, the Watchdog reset status flag is set and the associated WDRESETREQ reset request output pin is asserted. The external reset source is then responsible for resetting all or part of the Cortex-A9 MPCore design.</p>

Register ([mpcore](#)) Watchdog_Control_Register

Name	Watchdog_Control_Register
Software Name	WDT_CONTROL
Relative Address	0x00000628
Absolute Address	0xF8F00628
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Watchdog Control Register

Register Watchdog_Control_Register Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	rw	0x0	Reserved.
Prescaler (PRESCALER)	15:8	rw	0x0	The prescaler modifies the clock period for the decrementing event for the Counter Register.
	7:4	rw	0x0	Reserved.
Watchdog_mode (WD_MODE)	3	rw	0x0	1'b0 = Timer mode, default Writing a zero to this bit has no effect. You must use the Watchdog Disable Register to put the watchdog into timer mode. See Watchdog Disable Register on page 4-9. 1'b1 = Watchdog mode.
IT_Enable (IT_ENABLE)	2	rw	0x0	If set, the interrupt ID 30 is set as pending in the Interrupt Distributor when the event flag is set in the watchdog Status Register. In watchdog mode this bit is ignored
Auto_reload (AUTO_RELOAD)	1	rw	0x0	1'b0 = Single shot mode. Counter decrements down to zero, sets the event flag and stops. 1'b1 = Auto-reload mode. Each time the Counter Register reaches zero, it is reloaded with the value contained in the Load Register and then continues decrementing.
Watchdog_Enable (WD_ENABLE)	0	rw	0x0	Global watchdog enable 1'b0 = Watchdog is disabled and the counter does not decrement. All registers can still be read and /or written 1'b1 = Watchdog is enabled and the counter decrements normally.

Register ([mpcore](#)) Watchdog_Interrupt_Status_Register

Name	Watchdog_Interrupt_Status_Register
Software Name	WDT_ISR
Relative Address	0x0000062C
Absolute Address	0xF8F0062C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Watchdog Interrupt Status Register

Register Watchdog_Interrupt_Status_Register Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved
Event_flag (EVENT_FLAG)	0	rw	0x0	The event flag is a sticky bit that is automatically set when the Counter Register reaches zero in timer mode. If the watchdog interrupt is enabled, Interrupt ID 30 is set as pending in the Interrupt Distributor after the event flag is set. The event flag is cleared when written with a value of 1. Trying to write a zero to the event flag or a one when it is not set has no effect.

Register ([mpcore](#)) Watchdog_Reset_Status_Register

Name	Watchdog_Reset_Status_Register
Software Name	WDT_RST_STS
Relative Address	0x00000630
Absolute Address	0xF8F00630
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Watchdog Reset Status Register

Register Watchdog_Reset_Status_Register Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved. Writes are ignored, read data is always zero.
Reset_flag (RESET_FLAG)	0	rw	0x0	<p>The reset flag is a sticky bit that is automatically set when the Counter Register reaches zero and a reset request is sent accordingly. (In watchdog mode)</p> <p>The reset flag is cleared when written with a value of 1. Trying to write a zero to the reset flag or a one when it is not set has no effect. This flag is not reset by normal</p> <p>Cortex-A9 processor resets but has its own reset line, nWDRESET. nWDRESET must not be asserted when the Cortex-A9 processor reset assertion is the result of a watchdog reset request with WDRESETREQ. This distinction enables software to differentiate between a normal boot sequence, reset flag is zero, and one caused by a previous watchdog time-out, reset flag set to one.</p>

Register ([mpcore](#)) Watchdog_Disable_Register

Name	Watchdog_Disable_Register
Software Name	WDT_DISABLE
Relative Address	0x00000634
Absolute Address	0xF8F00634
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Watchdog Disable Register

Register Watchdog_Disable_Register Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x0	<p>Watchdog Disable Register</p> <p>Use the Watchdog Disable Register to switch from watchdog to timer mode. The software must write 0x12345678 then 0x87654321 successively to the Watchdog Disable Register so that the watchdog mode bit in the Watchdog Control Register is set to zero.</p> <p>If one of the values written to the Watchdog Disable Register is incorrect or if any other write occurs in between the two word writes, the watchdog remains in its current state.</p> <p>To reactivate the Watchdog, the software must write 1 to the watchdog mode bit of the Watchdog Control Register.</p>

Register ([mpcore](#)) ICDDCR

Name	ICDDCR
Software Name	GIC_DIST_EN
Relative Address	0x00001000
Absolute Address	0xF8F01000
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Distributor Control Register

Register ICDDCR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:2	rw	0x0	Reserved. Writes are ignored, read data is always zero.

Field Name	Bits	Type	Reset Value	Description
Enable_Non_secure	1	rw	0x0	<p>0 = disables all Non-secure interrupts control bits in the distributor from changing state because of any external stimulus change that occurs on the corresponding SPI or PPI signals</p> <p>1 = enables the distributor to update register locations for Non-secure interrupts</p> <p>FOR: ICDDCR_for_Non_secure_mode</p> <p>31,1 --> Reserved. Writes are ignored, read data is always zero.</p>
Enable_secure (GIC_EN_INT)	0	rw	0x0	<p>0 = disables all Secure interrupt control bits in the distributor from changing state because of any external stimulus change that occurs on the corresponding SPI or PPI signals.</p> <p>1 = enables the distributor to update register locations for Secure interrupts.</p> <p>FOR: ICDDCR_for_Non_secure_mode</p> <p>0 --> Enable_Non_secure --> 0 = disables all Non-secure interrupts control bits in the distributor from changing state because of any external stimulus change that occurs on the corresponding SPI or PPI signals</p> <p>1 = enables the distributor to update register locations for Non-secure interrupts</p>

Register ([mpcore](#)) ICIDICTR

Name	ICIDICTR
Software Name	GIC_IC_TYPE
Relative Address	0x00001004
Absolute Address	0xF8F01004
Width	32 bits
Access Type	ro
Reset Value	0x00000C22
Description	Interrupt Controller Type Register

Register ICDICTR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:29	ro	0x0	Reserved. Writes are ignored, read data is always zero.
LSPI (GIC_LSPI)	15:11	ro	0x1	Returns the number of Lockable Shared Peripheral Interrupts (LSPIs) that the controller contains. The encoding is: b11111 = 31 LSPIs, which are the interrupts of IDs 32-62. When CFGSDISABLE is HIGH then the interrupt controller prevents writes to any register locations that control the operating state of an LSPI.
SecurityExtn (GIC_DOMAIN)	10	ro	0x1	Returns the number of security domains that the controller contains: 1 = the controller contains two security domains. This bit always returns the value one.
SBZ	9:8	ro	0x0	Reserved
CPU_Number (GIC_CPU_NUM)	7:5	ro	0x1	The encoding is: b000 the Cortex-A9 MPCore configuration contains one Cortex-A9 processor. b001 the Cortex-A9 MPCore configuration contains two Cortex-A9 processors. b010 the Cortex-A9 MPCore configuration contains three Cortex-A9 processors. b011 the Cortex-A9 MPCore configuration contains four Cortex-A9 processors. b1xx: Unused values.
IT_Lines_Number (GIC_NUM_INT)	4:0	ro	0x2	The encoding is: b00000 = the distributor provides 32 interrupts, no external interrupt lines. b00001 = the distributor provides 64 interrupts, 32 external interrupt lines. b00010 = the distributor provides 96 interrupts, 64 external interrupt lines. b00011 = the distributor provide 128 interrupts, 96 external interrupt lines. b00100 = the distributor provides 160 interrupts, 128 external interrupt lines. b00101 = the distributor provides 192 interrupts, 160 external interrupt lines. b00110 = the distributor provides 224 interrupts, 192 external interrupt lines. b00111 = the distributor provides 256 interrupts, 224 external interrupt lines. All other values not used.

Register ([mpcore](#)) ICDIHDR

Name	ICDIHDR
Software Name	GIC_DIST_IDENT
Relative Address	0x00001008
Absolute Address	0xF8F01008
Width	32 bits
Access Type	ro
Reset Value	0x0102043B
Description	Distributor Implementer Identification Register

Register ICDIHDR Details

Field Name	Bits	Type	Reset Value	Description
Implementation_Version	31:24	ro	0x1	Gives implementation version number
Revision_Number (GIC_REV)	23:12	ro	0x20	Return the revision number of the controller
Implementer (GIC_IMPL)	11:0	ro	0x43B	Implementer Number

Register ([mpcore](#)) ICDISR0

Name	ICDISR0
Software Name	GIC_SECURITY0
Relative Address	0x00001080
Absolute Address	0xF8F01080
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Interrupt Security Register_0

Note: This register is the first in an array of 3 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
ICDISR0	0xf8f01080
ICDISR1	0xf8f01084
ICDISR2	0xf8f01088

Register ICDISR0 to ICDISR2 Details

Field Name	Bits	Type	Reset Value	Description
Security_Status (GIC_INT_NS)	31:0	rw	0x0	<p>The ICDISRn provide a Security status bit for each interrupt supported by the GIC.</p> <p>Each bit controls the security status of the corresponding interrupt.</p> <p>Accessible by Secure accesses only.</p> <p>The register addresses are RAZ/WI to Non-secure accesses.</p> <p>ICDISR0 is banked for each connected processor.</p>

Register ([mpcore](#)) ICDISER0

Name	ICDISER0
Software Name	GIC_ENABLE_SET
Relative Address	0x00001100
Absolute Address	0xF8F01100
Width	32 bits
Access Type	rw
Reset Value	0x0000FFFF
Description	Interrupt Set-enable Register 0

Register ICDISER0 Details

Field Name	Bits	Type	Reset Value	Description
Set (GIC_INT_EN)	31:0	rw	0xFFFF	<p>The ICDISERs provide a Set-enable bit for each interrupt supported by the GIC.</p> <p>Writing 1 to a Set-enable bit enables forwarding of the corresponding interrupt to the CPU interfaces.</p> <p>A register bit that corresponds to a Secure interrupt is RAZ/WI to Non-secure access.</p> <p>ICDISER0 is banked for each connected processor.</p>

Register ([mpcore](#)) ICDISER1

Name	ICDISER1
Relative Address	0x00001104
Absolute Address	0xF8F01104
Width	32 bits
Access Type	rw

Reset Value 0x00000000

Description Interrupt Set-enable Register 1

Register ICDISER1 Details

Field Name	Bits	Type	Reset Value	Description
Set	31:0	rw	0x0	The ICDISERs provide a Set-enable bit for each interrupt supported by the GIC. Writing 1 to a Set-enable bit enables forwarding of the corresponding interrupt to the CPU interfaces. A register bit that corresponds to a Secure interrupt is RAZ/WI to Non-secure access.

Register ([mpcore](#)) ICDISER2

Name ICDISER2

Relative Address 0x00001108

Absolute Address 0xF8F01108

Width 32 bits

Access Type rw

Reset Value 0x00000000

Description Interrupt Set-enable Register 2

Register ICDISER2 Details

Field Name	Bits	Type	Reset Value	Description
Set	31:0	rw	0x0	The ICDISERs provide a Set-enable bit for each interrupt supported by the GIC. Writing 1 to a Set-enable bit enables forwarding of the corresponding interrupt to the CPU interfaces. A register bit that corresponds to a Secure interrupt is RAZ/WI to Non-secure access.

Register ([mpcore](#)) ICDICER0

Name ICDICER0

Software Name GIC_DISABLE

Relative Address 0x00001180

Absolute Address 0xF8F01180

Width 32 bits

Access Type rw

Reset Value 0x0000FFFF

Description Interrupt Clear-Enable Register 0

Register ICDICER0 Details

Field Name	Bits	Type	Reset Value	Description
Clear (GIC_INT_CLR)	31:0	rw	0xFFFF	<p>The ICDICERs provide a Clear-enable bit for each interrupt supported by the GIC.</p> <p>Writing 1 to a Clear-enable bit disables forwarding of the corresponding interrupt to the CPU interfaces.</p> <p>A register bit that corresponds to a Secure interrupt is RAZ/WI to Non-secure accesses.</p> <p>ICDICER0 is banked for each connected processor.</p>

Register ([mpcore](#)) ICDICER1

Name ICDICER1

Relative Address 0x00001184

Absolute Address 0xF8F01184

Width 32 bits

Access Type rw

Reset Value 0x00000000

Description Interrupt Clear-Enable Register 1

Register ICDICER1 Details

Field Name	Bits	Type	Reset Value	Description
Clear	31:0	rw	0x0	<p>The ICDICERs provide a Clear-enable bit for each interrupt supported by the GIC.</p> <p>Writing 1 to a Clear-enable bit disables forwarding of the corresponding interrupt to the CPU interfaces.</p> <p>A register bit that corresponds to a Secure interrupt is RAZ/WI to Non-secure accesses.</p>

Register ([mpcore](#)) ICDICER2

Name ICDICER2

Relative Address 0x00001188

Absolute Address 0xF8F01188

Width 32 bits

Access Type	rw
Reset Value	0x00000000
Description	Interrupt Clear-Enable Register 2

Register ICDICER2 Details

Field Name	Bits	Type	Reset Value	Description
Clear	31:0	rw	0x0	<p>The ICDICERs provide a Clear-enable bit for each interrupt supported by the GIC.</p> <p>Writing 1 to a Clear-enable bit disables forwarding of the corresponding interrupt to the CPU interfaces.</p> <p>A register bit that corresponds to a Secure interrupt is RAZ/WI to Non-secure accesses.</p>

Register ([mpcore](#)) ICDISPR0

Name	ICDISPR0
Software Name	GIC_PENDING_SET0
Relative Address	0x00001200
Absolute Address	0xF8F01200
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Interrupt Set-pending Register_0

Note: This register is the first in an array of 3 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
ICDISPR0	0xf8f01200
ICDISPR1	0xf8f01204
ICDISPR2	0xf8f01208

Register ICDISPR0 to ICDISPR2 Details

Field Name	Bits	Type	Reset Value	Description
Set (GIC_PEND_SET)	31:0	rw	0x0	<p>The ICDISPRs provide a Set-pending bit for each interrupt supported by the GIC.</p> <p>Writing 1 to a Set-pending bit sets the status of the corresponding peripheral interrupt to pending.</p> <p>A register bit that corresponds to a Secure interrupt is RAZ/WI to Non-secure accesses.</p> <p>ICDISPR0 is banked for each connected processor.</p>

Register ([mpcore](#)) ICDICPR0

Name	ICDICPR0
Software Name	GIC_PENDING_CLR0
Relative Address	0x00001280
Absolute Address	0xF8F01280
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Interrupt Clear-Pending Register_0

Note: This register is the first in an array of 3 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
ICDICPR0	0xf8f01280
ICDICPR1	0xf8f01284
ICDICPR2	0xf8f01288

Register ICDICPR0 to ICDICPR2 Details

Field Name	Bits	Type	Reset Value	Description
Clear (GIC_PEND_CLR)	31:0	rw	0x0	<p>The ICDICPRs provide a Clear-pending bit for each interrupt supported by the GIC.</p> <p>Writing 1 to a Clear-pending bit clears the status of the corresponding peripheral interrupt to pending.</p> <p>A register bit that corresponds to a Secure interrupt is RAZ/WI to Non-secure accesses.</p> <p>ICDICPR0 is banked for each connected processor.</p>

Register ([mpcore](#)) ICDABR0

Name	ICDABR0
Software Name	GIC_ACTIVE0
Relative Address	0x00001300
Absolute Address	0xF8F01300
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Active Bit register_0

Note: This register is the first in an array of 3 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
ICDABR0	0xf8f01300
ICDABR1	0xf8f01304
ICDABR2	0xf8f01308

Register ICDABR0 to ICDABR2 Details

Field Name	Bits	Type	Reset Value	Description
Active (MASK)	31:0	rw	0x0	<p>The ICDABRs provide an Active bit for each interrupt supported by the GIC.</p> <p>The bit reads as one if the status of the interrupt is active or active and pending.</p> <p>Read the ICDSR or ICDCPR to find the pending status of the interrupt.</p> <p>ICDABR0 is banked for each connected processor.</p>

Register ([mpcore](#)) ICDIPR0

Name	ICDIPR0
Software Name	GIC_PRIORITY0
Relative Address	0x00001400
Absolute Address	0xF8F01400
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Interrupt Priority Register_0

Note: This register is the first in an array of 24 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
ICDIPR0	0xf8f01400
ICDIPR1	0xf8f01404
ICDIPR2	0xf8f01408
ICDIPR3	0xf8f0140c
ICDIPR4	0xf8f01410
ICDIPR5	0xf8f01414
ICDIPR6	0xf8f01418
ICDIPR7	0xf8f0141c
ICDIPR8	0xf8f01420
ICDIPR9	0xf8f01424
ICDIPR10	0xf8f01428
ICDIPR11	0xf8f0142c
ICDIPR12	0xf8f01430
ICDIPR13	0xf8f01434
ICDIPR14	0xf8f01438
ICDIPR15	0xf8f0143c
ICDIPR16	0xf8f01440
ICDIPR17	0xf8f01444
ICDIPR18	0xf8f01448
ICDIPR19	0xf8f0144c
ICDIPR20	0xf8f01450
ICDIPR21	0xf8f01454
ICDIPR22	0xf8f01458
ICDIPR23	0xf8f0145c

Register ICDIPR0 to ICDIPR23 Details

Field Name	Bits	Type	Reset Value	Description
Priority (MASK)	31:0	rw	0x0	<p>The ICDIPRs provide an 8-bit Priority field for each interrupt supported by the GIC.</p> <p>These registers are byte accessible.</p> <p>A register field that corresponds to a Secure interrupt is RAZ/WI to Non-secure accesses.</p> <p>ICDIPR0 to ICDIPR7 are banked for each connected processor</p>

Register ([mpcore](#)) ICDIPTR0

Name	ICDIPTR0
Software Name	GIC_SPI_TARGET0
Relative Address	0x00001800
Absolute Address	0xF8F01800
Width	32 bits
Access Type	rw
Reset Value	0x01010101
Description	Interrupt Processor Targets Register_0

Note: This register is the first in an array of 24 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
ICDIPTR0	0xf8f01800
ICDIPTR1	0xf8f01804
ICDIPTR2	0xf8f01808
ICDIPTR3	0xf8f0180c
ICDIPTR4	0xf8f01810
ICDIPTR5	0xf8f01814
ICDIPTR6	0xf8f01818
ICDIPTR7	0xf8f0181c
ICDIPTR8	0xf8f01820
ICDIPTR9	0xf8f01824
ICDIPTR10	0xf8f01828
ICDIPTR11	0xf8f0182c
ICDIPTR12	0xf8f01830
ICDIPTR13	0xf8f01834
ICDIPTR14	0xf8f01838
ICDIPTR15	0xf8f0183c
ICDIPTR16	0xf8f01840
ICDIPTR17	0xf8f01844
ICDIPTR18	0xf8f01848
ICDIPTR19	0xf8f0184c
ICDIPTR20	0xf8f01850
ICDIPTR21	0xf8f01854

Name	Address
ICDIPTR22	0xf8f01858
ICDIPTR23	0xf8f0185c

Register ICDIPTR0 to ICDIPTR23 Details

Field Name	Bits	Type	Reset Value	Description
Target (GIC_SPI_CPU _n)	31:0	rw	0x1010101	<p>The ICDIPTRs provide an 8-bit CPU targets field for each interrupt supported by the GIC.</p> <p>This field stores the lists of processors that the interrupt is sent to if it is asserted.</p> <p>These registers are byte accessible.</p> <p>A register field that corresponds to a Secure interrupt is RAZ/WI to Non-secure accesses.</p> <p>ICDIPTR0 to ICDIPTR7 are read-only, and each field returns a value corresponding only to the processor reading the register.</p> <p>Meaning of CPU targets field bit values:</p> <p>0bxxxxxx1: CPU interface 0</p> <p>0bxxxxxx1x: CPU interface 1</p>

Register ([mpcore](#)) ICDICFR0

Name	ICDICFR0
Software Name	GIC_INT_CFG
Relative Address	0x00001C00
Absolute Address	0xF8F01C00
Width	32 bits
Access Type	rw
Reset Value	0xAAAAAAAA
Description	Interrupt Configuration Register 0

Register ICDICFR0 Details

Field Name	Bits	Type	Reset Value	Description
Config (MASK)	31:0	rw	0xAAAAAAAAA	<p>The ICDICFRs provide a 2-bit Int_config field for each interrupt supported by the GIC.</p> <p>A register field that corresponds to Secure interrupt is RAZ/WI to Non-secure accesses.</p> <p>ICDICFR0 is read-only.</p> <p>Meaning of each 2-bit Int_config:</p> <p>Sgi: both bits are read-only; always 0b10</p> <p>PPI: both bits are read-only; always 0b01 (active LOW level sensitive) for PPI[1], and [4]; always 0b11(rising-edge sensitive) for PPI[0], [2], and [3].</p> <p>SPI: LSB is read-only, and is always 1;</p> <p>0b01=active HIGH level sensitive;</p> <p>0b11=rising-edge sensitive.</p>

Register ([mpcore](#)) ICDICFR1

Name	ICDICFR1
Relative Address	0x00001C04
Absolute Address	0xF8F01C04
Width	32 bits
Access Type	rw
Reset Value	0x7DC00000
Description	Interrupt Configuration Register 1

Register ICDICFR1 Details

Field Name	Bits	Type	Reset Value	Description
Config	31:0	rw	0x7DC00000	<p>(ditto)</p> <p>ICDICFR1 is banked for each connected processor.</p>

Register ([mpcore](#)) ICDICFR2

Name	ICDICFR2
Relative Address	0x00001C08
Absolute Address	0xF8F01C08
Width	32 bits
Access Type	rw
Reset Value	0x55555555

Description Interrupt Configuration Register 2

Register ICDICFR2 Details

Field Name	Bits	Type	Reset Value	Description
Config	31:0	rw	0x55555555	(ditto)

Register ([mpcore](#)) ICDICFR3

Name ICDICFR3
Relative Address 0x00001C0C
Absolute Address 0xF8F01C0C
Width 32 bits
Access Type rw
Reset Value 0x55555555
Description Interrupt Configuration Register 3

Register ICDICFR3 Details

Field Name	Bits	Type	Reset Value	Description
Config	31:0	rw	0x55555555	(ditto)

Register ([mpcore](#)) ICDICFR4

Name ICDICFR4
Relative Address 0x00001C10
Absolute Address 0xF8F01C10
Width 32 bits
Access Type rw
Reset Value 0x55555555
Description Interrupt Configuration Register 4

Register ICDICFR4 Details

Field Name	Bits	Type	Reset Value	Description
Config	31:0	rw	0x55555555	(ditto)

Register ([mpcore](#)) ICDICFR5

Name ICDICFR5

Relative Address	0x00001C14
Absolute Address	0xF8F01C14
Width	32 bits
Access Type	rw
Reset Value	0x55555555
Description	Interrupt Configuration Register 5

Register ICDICFR5 Details

Field Name	Bits	Type	Reset Value	Description
Config	31:0	rw	0x55555555	(ditto)

Register ([mpcore](#)) ppi_status

Name	ppi_status
Software Name	GIC_PPI_STAT
Relative Address	0x00001D00
Absolute Address	0xF8F01D00
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	PPI Status Register

Register ppi_status Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved. Writes are ignored, read data is always zero
ppi_status	15:11	ro	0x0	Returns the status of the PPI(4:0) inputs on the distributor: * PPI[4] is nIRQ * PPI[3] is the private watchdog * PPI[2] is the private timer * PPI[1] is nFIQ * PPI[0] is the global timer. PPI[1] and PPI[4] are active LOW PPI[0], PPI[2] and PPI[3] are active HIGH. Note These bits return the actual status of the PPI(4:0) signals. The ICDISPRn and ICDICPRn registers can also provide the PPI(4:0) status but because you can write to these registers then they might not contain the actual status of the PPI(4:0) signals.
SBZ	10:0	ro	0x0	SBZ

Register ([mpcore](#)) spi_status_0

Name	spi_status_0
Software Name	GIC_SPI_STAT
Relative Address	0x00001D04
Absolute Address	0xF8F01D04
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	SPI Status Register 0

Register spi_status_0 Details

Field Name	Bits	Type	Reset Value	Description
spi_status (GIC_SPI_N)	31:0	ro	0x0	Returns the status of the IRQ ID32 to ID63 inputs on the distributor. These bits return the actual status of the IRQ signals. Note: The ICDISPR1 and ICDICPR1 registers can also provide the IRQ status but because you can write to these registers then they might not contain the actual status of the IRQ signals.

Register ([mpcore](#)) spi_status_1

Name	spi_status_1
Relative Address	0x00001D08
Absolute Address	0xF8F01D08
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	SPI Status Register 1

Register spi_status_1 Details

Field Name	Bits	Type	Reset Value	Description
spi_status (GIC_SPI_N)	31:0	ro	0x0	Returns the status of the IRQ ID64 to ID95 inputs on the distributor. These bits return the actual status of the IRQ signals. Note: The ICDISPR2 and ICDICPR2 registers can also provide the IRQ status but because you can write to these registers then they might not contain the actual status of the IRQ signals.

Register ([mpcore](#)) ICDSGIR

Name	ICDSGIR
Software Name	GIC_SFI_TRIG
Relative Address	0x00001F00
Absolute Address	0xF8F01F00
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Software Generated Interrupt Register

Register ICDSGIR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	rw	0x0	reserved
TargetListFilter (TRGFILT)	25:24	rw	0x0	0b00: send the interrupt to the CPU interfaces specified in the CPUTargetList field 0b01: send the interrupt to all CPU interfaces except the CPU interface that requested the interrupt 0b10: send the interrupt on only to the CPU interface that requested the interrupt 0b11: reserved
CPUTargetList (CPU)	23:16	rw	0x0	When TargetListFilter is 0b00, defines the CPU interfaces the Distributor must send the interrupt to. Each bit refers to the corresponding CPU interface.
SATT	15	rw	0x0	Determines the condition for sending the SGI specified in the SGIINTID field to a specified CPU interfaces: 0: only if the SGI is configured as Secure on that interface. 1: only if the SGI is configured as Non-secure on that interface.
SBZ	14:4	rw	0x0	SBZ
SGIINTID (INTID)	3:0	rw	0x0	The Interrupt ID of the SGI to send to the specified CPU interfaces.

Register ([mpcore](#)) ICPIDR4

Name	ICPIDR4
Software Name	GIC_PERPHID
Relative Address	0x00001FD0
Absolute Address	0xF8F01FD0
Width	32 bits
Access Type	rw
Reset Value	0x00000004
Description	Peripheral ID4

Register ICPIDR4 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:4	rw	0x0	reserved
ContinuationCode	3:0	rw	0x4	ARM-defined ContinuationCode field

Register ([mpcore](#)) ICPIDR5

Name	ICPIDR5
Relative Address	0x00001FD4
Absolute Address	0xF8F01FD4
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Peripheral ID5

Register ICPIDR5 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:0	rw	0x0	Reserved

Register ([mpcore](#)) ICPIDR6

Name	ICPIDR6
Relative Address	0x00001FD8
Absolute Address	0xF8F01FD8
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Peripheral ID6

Register ICPIDR6 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:0	rw	0x0	Reserved

Register ([mpcore](#)) ICPIDR7

Name	ICPIDR7
------	---------

Relative Address	0x00001FDC
Absolute Address	0xF8F01FDC
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Peripheral ID7

Register ICPIDR7 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:0	rw	0x0	Reserved

Register ([mpcore](#)) ICPIDR0

Name	ICPIDR0
Relative Address	0x00001FE0
Absolute Address	0xF8F01FE0
Width	32 bits
Access Type	rw
Reset Value	0x00000090
Description	Peripheral ID0

Register ICPIDR0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rw	0x0	reserved
DevID_low	7:0	rw	0x90	ARM-defined DevID[7:0] field

Register ([mpcore](#)) ICPIDR1

Name	ICPIDR1
Relative Address	0x00001FE4
Absolute Address	0xF8F01FE4
Width	32 bits
Access Type	rw
Reset Value	0x000000B3
Description	Peripheral ID1

Register ICPIDR1 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rw	0x0	reserved
ARChID_low	7:4	rw	0xB	ARM-defined ArchID[3:0] field
DevID_high	3:0	rw	0x3	ARM-defined DevID[11:8] field

Register ([mpcore](#)) ICPIDR2

Name	ICPIDR2
Relative Address	0x00001FE8
Absolute Address	0xF8F01FE8
Width	32 bits
Access Type	rw
Reset Value	0x0000001B
Description	Peripheral ID2

Register ICPIDR2 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rw	0x0	reserved
ArchRev	7:4	rw	0x1	ARM-defined ArchRev field
UsesJEPcode	3	rw	0x1	ARM-defined ContinuationCode field
ArchID_high	2:0	rw	0x3	ARM-defined ArchID[6:4] field

Register ([mpcore](#)) ICPIDR3

Name	ICPIDR3
Relative Address	0x00001FEC
Absolute Address	0xF8F01FEC
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Peripheral ID3

Register ICPIDR3 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rw	0x0	reserved
Revision	7:4	rw	0x0	ARM-defined Revision field
reserved	3:0	rw	0x0	reserved

Register ([mpcore](#)) ICCIDR0

Name	ICCIDR0
Software Name	GIC_PCELLID
Relative Address	0x00001FF0
Absolute Address	0xF8F01FF0
Width	32 bits
Access Type	rw
Reset Value	0x0000000D
Description	Component ID0

Register ICCIDR0 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0xD	ARM-defined fixed values for the preamble for component discovery

Register ([mpcore](#)) ICCIDR1

Name	ICCIDR1
Relative Address	0x00001FF4
Absolute Address	0xF8F01FF4
Width	32 bits
Access Type	rw
Reset Value	0x000000F0
Description	Component ID1

Register ICCIDR1 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0xF0	ARM-defined fixed values for the preamble for component discovery

Register ([mpcore](#)) ICCIDR2

Name	ICCIDR2
Relative Address	0x00001FF8
Absolute Address	0xF8F01FF8
Width	32 bits
Access Type	rw
Reset Value	0x00000005
Description	Component ID2

Register ICCIDR2 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x5	ARM-defined fixed values for the preamble for component discovery

Register ([mpcore](#)) ICCIDR3

Name	ICCIDR3
Relative Address	0x00001FFC
Absolute Address	0xF8F01FFC
Width	32 bits
Access Type	rw
Reset Value	0x000000B1
Description	Component ID3

Register ICCIDR3 Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0xB1	ARM-defined fixed values for the preamble for component discovery

B.25 On-Chip Memory (ocm)

Module Name	On-Chip Memory (ocm)
Base Address	0xF800C000 ocm
Description	On-Chip Memory Registers
Version	1.0
Doc Version	1.0
Vendor Info	Xilinx

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
OCM_PARITY_CTRL	0x00000000	32	rw	0x00000000	Control fields for RAM parity operation
OCM_PARITY_ERR_ADDR	0x00000004	32	rw	0x00000000	Stores the first parity error access address. This register is sticky and will retain its value unless explicitly cleared (written with 1's) with an APB write access. The physical RAM address is logged.
OCM_IRQ_STS	0x00000008	32	rw	0x00000000	Status of OCM Interrupt
OCM_CONTROL	0x0000000C	32	rw	0x00000000	Control fields for OCM

Register ([ocm](#)) OCM_PARITY_CTRL

Name	OCM_PARITY_CTRL
Relative Address	0x00000000
Absolute Address	0xF800C000
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Control fields for RAM parity operation

Register OCM_PARITY_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:21	rw	0x0	Returns 0 when read
OddParityEn	20:5	rw	0x0	Enable RAM Odd Parity Generation. One control bit per data byte (OddParity[0] controls Data[7:0] e.t.c) 0 - Even Parity generated 1 - Odd Parity generated
LockFailErrIrqEn	4	rw	0x0	Enable interrupt when an AXI LOCK ("locked access") command is detected.
MultipleParityErrIrqEn	3	rw	0x0	Same as SingleParityErrIrqEn, but enables IRQ on multiple parity errors detected. 0 - IRQ is not generated when parity error detected and ParityCheckDis=0 1 - IRQ is generated when parity error detected and ParityCheckDis=0.
SingleParityErrIrqEn	2	rw	0x0	Enable interrupt when a single parity error is detected. Note that even if this field is 0, the OCM_IRQ_STS register will still log the error if ParityCheckDis=0. This allows software the option of polling if an error occurred. 0 - IRQ is not generated when parity error detected and ParityCheckDis=0 1 - IRQ is generated when parity error detected and ParityCheckDis=0.
RdRespParityErrEn	1	rw	0x0	Enable AXI read 'SLVERR' response for parity error detection. 0 - Error will not be sent on AXI read channel when parity error detected 1 - Error will be sent on AXI read channel when parity error detected and ParityCheckDis=0
ParityCheckDis	0	rw	0x0	Disable RAM Parity Checking. No checking or logging of status will occur when 1. 0 - RAM Parity checking is enabled 1 - RAM Parity checking is disabled

Register ([ocm](#)) OCM_PARITY_ERRADDRESS

Name	OCM_PARITY_ERRADDRESS
Relative Address	0x00000004
Absolute Address	0xF800C004
Width	32 bits

Access Type	rw
Reset Value	0x00000000
Description	Stores the first parity error access address. This register is sticky and will retain its value unless explicitly cleared (written with 1's) with an APB write access. The physical RAM address is logged.

Register OCM_PARITY_ERRADDRESS Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	Return 0 when read
ParityErrAddress	13:0	rw	0x0	When a parity Error occurs, the access address associated with the error is logged here. The first error address will be held if multiple parity errors occur. Need an explicit write of all '1's' to reset/clear this field.

Register ([ocm](#)) OCM_IRQ_STS

Name	OCM_IRQ_STS
Relative Address	0x00000008
Absolute Address	0xF800C008
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Status of OCM Interrupt

Register OCM_IRQ_STS Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:3	rw	0x0	Return 0 when read
LockFailErr	2	rw	0x0	When set (1), indicates that an AXI LOCK has been attempted (not supported by OCM). This is a sticky bit. Once set it can only be cleared by explicitly writing a 1 to this field. This field drives the interrupt pin. (Associated irq enable bit must be set)

Field Name	Bits	Type	Reset Value	Description
MultipleParityErr	1	rw	0x0	Status of OCM multiple parity error. This is a sticky bit. Once set it can only be cleared by explicitly writing a 1 to this field. This field drives the interrupt pin. (Associated irq enable bit must be set) 0 - Multiple OCM parity Errors have not occurred 1 - Multiple OCM parity Errors have occurred
SingleParityErr	0	rw	0x0	Status of OCM single parity error. This is a sticky bit. Once set it can only be cleared by explicitly writing a 1 to this field. This field drives the interrupt pin (Associated irq enable bit must be set) 0 - Single OCM parity Error has not occurred 1 - Single OCM parity Error has occurred

Register ([ocm](#)) OCM_CONTROL

Name	OCM_CONTROL
Relative Address	0x0000000C
Absolute Address	0xF800C00C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Control fields for OCM

Register OCM_CONTROL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:3	rw	0x0	Return 0 when read
ArbShareTopSwScuWr DIs	2	rw	0x0	Controls the arbitration to memory between the topswitch port and the scuWr port. 0 : The topsw and the scuWr porst share the memory bandwith - 50% each. 1 : The scuWr takes higher priority over the topswitch port (unless the ScuWrPriorityLo bit is set)
reserved	1	rw	0x0	Reserved. Do not modify.
ScuWrPriorityLo	0	rw	0x0	When set (1), changes the priority of the SCU write port to LOW from Medium

B.26 Quad-SPI Flash Controller (qspi)

Module Name	Quad-SPI Flash Controller (qspi)
Software Name	XQSPIPS
Base Address	0xE000D000 qspi
Description	LQSPI module Registers
Version	1.0
Doc Version	0.8, based on 11/01/10 Linear Quad-SPI Controller Design Specification document
Vendor Info	Xilinx lqspi

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
Config_reg	0x00000000	32	mixed	x	SPI configuration register
Intr_status_REG	0x00000004	32	mixed	0x00000004	SPI interrupt status register
Intrpt_en_REG	0x00000008	32	mixed	0x00000000	Interrupt Enable register. Write only, read as 0.
Intrpt_dis_REG	0x0000000C	32	mixed	0x00000000	Interrupt disable register. Write only, read as 0.
Intrpt_mask_REG	0x00000010	32	ro	0x00000000	Interrupt mask register
En_REG	0x00000014	32	mixed	0x00000000	SPI_Enable Register
Delay_REG	0x00000018	32	rw	0x00000000	Delay Register
TXD0	0x0000001C	32	wo	0x00000000	Transmit Data Register. Keyhole addresses for the Transmit data FIFO. See also TXD1-3.
Rx_data_REG	0x00000020	32	ro	0x00000000	Receive Data Register
Slave_Idle_count_REG	0x00000024	32	mixed	0x000000FF	Slave Idle Count Register
TX_thres_REG	0x00000028	32	rw	0x00000001	TX_FIFO Threshold Register
RX_thres_REG	0x0000002C	32	rw	0x00000001	RX_FIFO Threshold Register
GPIO	0x00000030	32	rw	0x00000001	General Purpose Inputs and Outputs Register for the Quad-SPI Controller core
LPBK_DLY_ADJ	0x00000038	32	rw	0x00000033	Loopback Master Clock Delay Adjustment Register

Register Name	Address	Width	Type	Reset Value	Description
TXD1	0x00000080	32	wo	0x00000000	Transmit Data Register. Keyhole addresses for the Transmit data FIFO.
TXD2	0x00000084	32	wo	0x00000000	Transmit Data Register. Keyhole addresses for the Transmit data FIFO.
TXD3	0x00000088	32	wo	0x00000000	Transmit Data Register. Keyhole addresses for the Transmit data FIFO.
LQSPI_CFG	0x000000A0	32	rw	x	Configuration Register specifically for the Linear Quad-SPI Controller
LQSPI_STS	0x000000A4	9	rw	0x00000000	Status Register specifically for the Linear Quad-SPI Controller
MOD_ID	0x000000FC	32	rw	0x01090101	Module Identification register

Register ([qspi](#)) Config_reg

Name	Config_reg
Software Name	CR
Relative Address	0x00000000
Absolute Address	0xE000D000
Width	32 bits
Access Type	mixed
Reset Value	x
Description	SPI configuration register

Register Config_reg Details

Field Name	Bits	Type	Reset Value	Description
leg_flgsh (IFMODE)	31	rw	0x1	Flash memory interface mode control: 0 = legacy SPI mode 1 = Flash memory interface mode This control is required to enable or disable automatic recognition of instruction bytes in the first byte of a transfer. If this mode is disabled, the core will operate in standard SPI mode, with no dual- or quad-bit input or output capability; the extended bits will be configured as inputs to prevent any driver contention on these pins. If enabled, flash memory interface instructions are automatically recognized and the I/O configured accordingly.
reserved	30:27	ro	0x0	Reserved, read as zero, ignored on write.
endian (ENDIAN)	26	rw	0x0	0 for little endian format when writing to the transmit data register 0x1C or reading from the receive data register 0x20. 1 for big endian format when writing to the transmit data register 0x1C or reading from the receive data register 0x20.
reserved	25:20	ro	0x0	Reserved, read as zero, ignored on write.
Holdb_dr	19	rw	0x0	Holdb and WPn pins are driven in normal/fast read or dual output/io read by the controller, if set, else external pull-high is required. Both pins are always driven by the controller in quad mode.
reserved	18	rw	0x0	Reserved
reserved	17	rw	x	Reserved, do not modify
Man_start_com (MANSTRT)	16	wo	0x0	Manual Start Command 1 = start transmission of data 0 = don't care
Man_start_en (MANSTRTEN)	15	rw	0x0	Manual Start Enable 1 = enables manual start 0 = auto mode
Manual_CS (SSFORCE)	14	rw	0x0	Manual CS 1 = manual CS mode 0 = auto mode
reserved	13:11	rw	0x0	Reserved
PCS	10	rw	0x0	Peripheral chip select line, directly drive n_ss_out if Manual_C is set

Field Name	Bits	Type	Reset Value	Description
reserved	9	rw	0x0	Reserved
REF_CLK	8	rw	0x0	Reserved. Must be 0
FIFO_WIDTH	7:6	rw	0x0	FIFO width Must be set to 2'b11 (32bits). All other settings are not supported.
BAUD_RATE_DIV	5:3	rw	0x0	Master mode baud rate divisor 000 = divide by 2 001 = divide by 4 010 = divide by 8 011 = divide by 16 100 = divide by 32 101 = divide by 64 110 = divide by 128 111 = divide by 256
CLK_PH (CPHA)	2	rw	0x0	Clock phase 1 = the SPI clock is inactive outside the word 0 = the SPI clock is active outside the word Note : For {CLK_PH, CLK_POL}, only 2'b11 and 2'b00 are supported.
CLK_POL (CPOL)	1	rw	0x0	Clock polarity outside SPI word 1 = the SPI clock is quiescent high 0 = the SPI clock is quiescent low Note : For {CLK_PH, CLK_POL}, only 2'b11 and 2'b00 are supported.
MODE_SEL (MSTREN)	0	rw	0x0	Mode select 1 = the SPI is in master mode 0 = the SPI is in slave mode

Register ([qspi](#)) Intr_status_REG

Name	Intr_status_REG
Software Name	SR
Relative Address	0x00000004
Absolute Address	0xE000D004
Width	32 bits
Access Type	mixed
Reset Value	0x00000004
Description	SPI interrupt status register

Register Intr_status_REG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:7	ro	0x0	Reserved, read as zero, ignored on write.
TX_FIFO_underflow (IXR_TXUF)	6	wtc	0x0	TX FIFO underflow, write one to this bit location to clear. 1 = underflow is detected 0 = no underflow has been detected Write 1 to this bit location to clear
RX_FIFO_full (IXR_RXFULL)	5	ro	0x0	RX FIFO full (current FIFO status) 1 = FIFO is full 0 = FIFO is not full
RX_FIFO_not_empty (IXR_RXNEMPTY)	4	ro	0x0	RX FIFO not empty (current FIFO status) 1 = FIFO has more than or equal to THRESHOLD entries 0 = FIFO has less than RX THRESHOLD entries
TX_FIFO_full (IXR_TXFULL)	3	ro	0x0	TX FIFO full (current FIFO status) 1 = FIFO is full 0 = FIFO is not full
TX_FIFO_not_full (IXR_TXOW)	2	ro	0x1	TX FIFO not full (current FIFO status) 1 = FIFO has less than THRESHOLD entries 0 = FIFO has more than or equal to THRESHOLD entries
reserved	1	ro	0x0	Reserved, read as zero, ignored on write.
RX_OVERFLOW (IXR_RXOVR)	0	wtc	0x0	Receive Overflow interrupt, write one to this bit location to clear. 1 = overflow occurred 0 = no overflow occurred Write 1 to this bit location to clear

Register ([qspi](#)) Intrpt_en_REG

Name	Intrpt_en_REG
Software Name	IER
Relative Address	0x00000008
Absolute Address	0xE000D008
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Interrupt Enable register. Write only, read as 0.

Register Intrpt_en_REG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:7	ro	0x0	Reserved, read as zero, ignored on write.
TX_FIFO_underflow (IXR_TXUF)	6	wo	0x0	TX FIFO underflow enable 1 = enable the interrupt 0 = no effect
RX_FIFO_full (IXR_RXFULL)	5	wo	0x0	RX FIFO full enable 1 = enable the interrupt 0 = no effect
RX_FIFO_not_empty (IXR_RXNEMPTY)	4	wo	0x0	RX FIFO not empty enable 1 = enable the interrupt 0 = no effect
TX_FIFO_full (IXR_TXFULL)	3	wo	0x0	TX FIFO full enable 1 = enable the interrupt 0 = no effect
TX_FIFO_not_full (IXR_TXOW)	2	wo	0x0	TX FIFO not full enable 1 = enable the interrupt 0 = no effect
reserved	1	wo	0x0	Reserved, read as zero, ignored on write.
RX_OVERFLOW (IXR_RXOVR)	0	wo	0x0	Receive Overflow interrupt enable 1 = enable the interrupt 0 = no effect

Register ([gspl](#)) Intrpt_dis_REG

Name	Intrpt_dis_REG
Software Name	IDR
Relative Address	0x0000000C
Absolute Address	0xE000D00C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Interrupt disable register. Write only, read as 0.

Register Intrpt_dis_REG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:7	ro	0x0	Reserved, read as zero, ignored on write.
TX_FIFO_underflow (IXR_TXUF)	6	wo	0x0	TX FIFO underflow enable 1 = disables the interrupt 0 = no effect
RX_FIFO_full (IXR_RXFULL)	5	wo	0x0	RX FIFO full enable 1 = disables the interrupt 0 = no effect
RX_FIFO_not_empty (IXR_RXNEMPTY)	4	wo	0x0	RX FIFO not empty enable 1 = disables the interrupt 0 = no effect
TX_FIFO_full (IXR_TXFULL)	3	wo	0x0	TX FIFO full enable 1 = disables the interrupt 0 = no effect
TX_FIFO_not_full (IXR_TXOW)	2	wo	0x0	TX FIFO not full enable 1 = disables the interrupt 0 = no effect
reserved	1	wo	0x0	Reserved
RX_OVERFLOW (IXR_RXOVR)	0	wo	0x0	Receive Overflow interrupt enable 1 = disables the interrupt 0 = no effect

Register ([gspl](#)) Intrpt_mask_REG

Name	Intrpt_mask_REG
Software Name	IMR
Relative Address	0x00000010
Absolute Address	0xE000D010
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Interrupt mask register

Register Intrpt_mask_REG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:7	ro	0x0	Reserved, read as zero, ignored on write.
TX_FIFO_underflow (IXR_TXUF)	6	ro	0x0	TX FIFO underflow enable 1 = interrupt is disabled 0 = interrupt is enabled
RX_FIFO_full (IXR_RXFULL)	5	ro	0x0	RX FIFO full enable 1 = interrupt is disabled 0 = interrupt is enabled
RX_FIFO_not_empty (IXR_RXNEMPTY)	4	ro	0x0	RX FIFO not empty enable 1 = interrupt is disabled 0 = interrupt is enabled
TX_FIFO_full (IXR_TXFULL)	3	ro	0x0	TX FIFO full enable 1 = interrupt is disabled 0 = interrupt is enabled
TX_FIFO_not_full (IXR_TXOW)	2	ro	0x0	TX FIFO not full enable 1 = interrupt is disabled 0 = interrupt is enabled
reserved	1	ro	0x0	Reserved
RX_OVERFLOW (IXR_RXOVR)	0	ro	0x0	Receive Overflow interrupt enable 1 = interrupt is disabled 0 = interrupt is enabled

Register ([gspl](#)) En_REG

Name	En_REG
Software Name	ER
Relative Address	0x00000014
Absolute Address	0xE000D014
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	SPI_Enable Register

Register En_REG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	ro	0x0	Reserved, read as zero, ignored on write.
SPI_EN (ENABLE)	0	rw	0x0	SPI_Enable 1 = enable the SPI 0 = disable the SPI

Register ([gspl](#)) Delay_REG

Name	Delay_REG
Software Name	DR
Relative Address	0x00000018
Absolute Address	0xE000D018
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Delay Register

Register Delay_REG Details

Field Name	Bits	Type	Reset Value	Description
d_nss	31:24	rw	0x0	Delay in SPI REFERENCE CLOCK or ext_clk cycles for the length that the master mode chip select outputs are de-asserted between words when cpha=0.
d_btwn (BTWN)	23:16	rw	0x0	Delay in SPI REFERENCE CLOCK or ext_clk cycles between one chip select being de-activated and the activation of another
d_after (AFTER)	15:8	rw	0x0	Delay in SPI REFERENCE CLOCK or ext_clk cycles between last bit of current word and the first bit of the next word.
d_int (INIT)	7:0	rw	0x0	Added delay in SPI REFERENCE CLOCK or ext_clk cycles between setting n_ss_out low and first bit transfer.

Register ([qspi](#)) TXD0

Name	TXD0
Software Name	TXD_00
Relative Address	0x0000001C
Absolute Address	0xE000D01C
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	Transmit Data Register. Keyhole addresses for the Transmit data FIFO. See also TXD1-3.

Register TXD0 Details

Field Name	Bits	Type	Reset Value	Description
TXD	31:0	wo	0x0	Data to TX FIFO, for 4-byte instruction for normal read/write data transfer.

Register ([qspi](#)) Rx_data_REG

Name	Rx_data_REG
Software Name	RXD
Relative Address	0x00000020
Absolute Address	0xE000D020
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Receive Data Register

Register Rx_data_REG Details

Field Name	Bits	Type	Reset Value	Description
RX_FIFO_data	31:0	ro	0x0	Data from TX FIFO

Register ([qspi](#)) Slave_Idle_count_REG

Name	Slave_Idle_count_REG
Software Name	SICR
Relative Address	0x00000024
Absolute Address	0xE000D024

Width	32 bits
Access Type	mixed
Reset Value	0x000000FF
Description	Slave Idle Count Register

Register Slave_Idle_count_REG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	ro	0x0	Reserved, read as zero, ignored on write.
Slave_Idle_coun (MASK)	7:0	rw	0xFF	SPI in slave mode detects a start only when the external SPI master serial clock (sclk_in) is stable (quiescent state) for SPI REFERENCE CLOCK cycles specified by slave idle count register or when the SPI is deselected.

Register ([gsqi](#)) TX_thres_REG

Name	TX_thres_REG
Software Name	TXWR
Relative Address	0x00000028
Absolute Address	0xE000D028
Width	32 bits
Access Type	rw
Reset Value	0x00000001
Description	TX_FIFO Threshold Register

Register TX_thres_REG Details

Field Name	Bits	Type	Reset Value	Description
DEPTH_of_TX_FIFO	31:0	rw	0x1	Defines the level at which the TX FIFO not full interrupt is generated

Register ([gsqi](#)) RX_thres_REG

Name	RX_thres_REG
Relative Address	0x0000002C
Absolute Address	0xE000D02C
Width	32 bits

Access Type rw

Reset Value 0x00000001

Description RX FIFO Threshold Register

Register RX_thres_REG Details

Field Name	Bits	Type	Reset Value	Description
DEPTH_of_RX_FIFO	31:0	rw	0x1	Defines the level at which the RX FIFO not empty interrupt is generated

Register ([qspi](#)) GPIO

Name GPIO

Relative Address 0x00000030

Absolute Address 0xE000D030

Width 32 bits

Access Type rw

Reset Value 0x00000001

Description General Purpose Inputs and Outputs Register for the Quad-SPI Controller core

Register GPIO Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved for future GPIO.
WP_N	0	rw	0x1	Write Protect. Write Protect output for flash devices supporting this function. Active low (may be inverted externally to the core if required for flash devices requiring active high write protect signal.)

Register ([qspi](#)) LPBK_DLY_ADJ

Name LPBK_DLY_ADJ

Relative Address 0x00000038

Absolute Address 0xE000D038

Width 32 bits

Access Type rw

Reset Value 0x00000033

Description Loopback Master Clock Delay Adjustment Register

Register LPBK_DLY_ADJ Details

Register for adjusting the internal loopback clock delay for read data capturing.

This feature is only active if bit 5 is set AND if the baud rate divisor in reg 0x00 is programmed to 2 (i.e., 000).

Field Name	Bits	Type	Reset Value	Description
reserved	31:9	rw	0x0	Reserved.
LPBK_SEL	8	rw	0x0	Set to 0, do not modify
LPBK_PH	7	rw	0x0	Set to 0, do not modify
reserved	6	rw	0x0	Set to 0, do not modify
USE_LPBK	5	rw	0x1	Use internal loopback master clock for read data capturing when baud rate divisor (reg 0x00) is 2
DLY1	4:3	rw	0x2	Must be set to 00 if Loopback clk used
DLY0	2:0	rw	0x3	Must be set to 00 if Loopback clk used

Register ([qspi](#)) TXD1

Name	TXD1
Software Name	TXD_01
Relative Address	0x00000080
Absolute Address	0xE000D080
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	Transmit Data Register. Keyhole addresses for the Transmit data FIFO.

Register TXD1 Details

Field Name	Bits	Type	Reset Value	Description
TXD	31:0	wo	0x0	Data to TX FIFO, for 1-byte instruction, not for normal data transfer. In little endian mode (default), only bits 7:0 are valid, bits 31:8 are ignored. In big endian mode, only the 8 MS bits are valid.

Register ([qspi](#)) TXD2

Name	TXD2
Software Name	TXD_10
Relative Address	0x00000084

Absolute Address	0xE000D084
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	Transmit Data Register. Keyhole addresses for the Transmit data FIFO.

Register TXD2 Details

Field Name	Bits	Type	Reset Value	Description
TXD	31:0	wo	0x0	Data to TX FIFO, for 2-byte instruction, not for normal data transfer. In little endian mode (default), only bits 15:0 are valid, bits 31:16 are ignored. In big endian mode, only the 16 MS bits are valid.

Register ([qspi](#)) TXD3

Name	TXD3
Software Name	TXD_11
Relative Address	0x00000088
Absolute Address	0xE000D088
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	Transmit Data Register. Keyhole addresses for the Transmit data FIFO.

Register TXD3 Details

Field Name	Bits	Type	Reset Value	Description
TXD	31:0	wo	0x0	Data to TX FIFO, for 3-byte instruction, not for normal data transfer. In little endian mode (default), only bits 23:0 are valid, bits 31:24 are ignored. In big endian mode, only the 24 MS bits are valid.

Register ([qspi](#)) LQSPI_CFG

Name	LQSPI_CFG
Software Name	LQSPI_CR
Relative Address	0x000000A0

Absolute Address	0xE000D0A0
Width	32 bits
Access Type	rw
Reset Value	x
Description	Configuration Register specifically for the Linear Quad-SPI Controller

Register LQSPI_CFG Details

Field Name	Bits	Type	Reset Value	Description
LQ_MODE (LINEAR)	31	rw	0x0	Linear quad SPI mode, if set, else quad SPI mode
TWO_MEM	30	rw	0x0	Both upper and lower memories are active, if set
SEP_BUS	29	rw	0x0	Separate memory bus, if set. Only has meaning if bit 30 is set
U_PAGE	28	rw	0x0	Upper memory page, if set. Only has meaning if bit 30 is set AND bit 29 is clear AND bit 31 is clear. In LQSPI mode, address bit 25 will indicate lower (0) or upper (1) page. In IO mode, this bit is used to select the lower or upper memory for configuration or read/write operations.
reserved	27	rw	0x0	Reserved
reserved	26	rw	x	Reserved. Write to this location may lead to unpredictable behavior.

Field Name	Bits	Type	Reset Value	Description
MODE_EN	25	rw	0x1	<p>Enable MODE_BITS[23:16] to be sent, if set.</p> <p>This bit MUST BE SET for dual I/O or quad I/O read (specified through [7:0]).</p> <p>This bit MUST BE CLEAR for all other read modes as they do not have mode bits.</p> <p>If this bit is 0, bits 24, and [23:16] are ignored.</p> <p>Here is a summary of how bits 25, 24 and 23:16 are related.</p> <p>Bit 25</p> <p>bit 24</p> <p>bits 23:16</p> <p>0</p> <p>x</p> <p>x</p> <p>1</p> <p>0</p> <p>not 8'bxx10xxxx</p> <p>1</p> <p>1</p> <p>8'bxx10xxxx</p>
MODE_ON	24	rw	0x1	<p>This bit is only relevant if bit 25 is set, else it is ignored.</p> <p>If this bit is set, instruction code is only sent for the very first read transfer.</p> <p>If this bit is clear, instruction code will be sent for all read transfers.</p> <p>This bit is configured in association with the MODE_BITS.</p> <p>For Winbond devices, this bit MUST BE SET if the MODE_BITS are 8'bxx10xxxx, else this bit MUST BE CLEAR.</p>
MODE_BITS	23:16	rw	0xA0	<p>These bits are only relevant if bit 25 is set, else it is ignored.</p> <p>If bit 25 is set, this value is required for both dual I/O read and quad I/O read.</p> <p>See vendor's datasheet for more information.</p> <p>For Winbond's device, the continuous read mode value is 8'bxx10xxxx to skip the instruction code for the next read transfer, else instruction code is sent for all read transfers.</p> <p>Bit 24 has to be configured accordingly with this value.</p>
reserved	15:11	rw	x	Reserved, value is undefined when read.

Field Name	Bits	Type	Reset Value	Description
DUMMY_BYTE (DUMMY)	10:8	rw	0x2	Number of dummy bytes between address and return read data
INST_CODE (INST)	7:0	rw	0xEB	Read instruction code. The known read instruction codes are: 8'h03 - Read 8'h0B - Fast read 8'h3B - Fast read dual output 8'h6B - Fast read quad output 8'hBB - Fast read dual I/O 8'hEB - Fast read quad I/O

Register ([qspi](#)) LQSPI_STS

Name	LQSPI_STS
Software Name	LQSPI_SR
Relative Address	0x000000A4
Absolute Address	0xE000D0A4
Width	9 bits
Access Type	rw
Reset Value	0x00000000
Description	Status Register specifically for the Linear Quad-SPI Controller

Register LQSPI_STS Details

Field Name	Bits	Type	Reset Value	Description
reserved	8:3	rw	0x0	Reserved
D_FSM_ERR (FB_RECVD)	2	rw	0x0	Data FSM error, if set
WR_RECVD	1	rw	0x0	AXI write command received, if set
reserved	0	rw	0x0	Reserved

Register ([qspi](#)) MOD_ID

Name	MOD_ID
Relative Address	0x000000FC
Absolute Address	0xE000D0FC
Width	32 bits
Access Type	rw

Reset Value 0x01090101

Description Module Identification register

Register MOD_ID Details

Field Name	Bits	Type	Reset Value	Description
	31:0	rw	0x1090101	Module ID value.

B.27 SD Controller (sdio)

Module Name	SD Controller (sdio)
Base Address	0xE0100000 sd0 0xE0101000 sd1
Description	SD2.0/ SDIO2.0/ MMC3.31 AHB Host ControllerRegisters Instance no. 0.
Version	4.4a
Doc Version	4.0
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
SDMA_system_addresses_register	0x00000000	32	rw	0x00000000	This register contains the system memory address for a DMA transfer. When the Host Controller (HC) stops a DMA transfer, this register shall point to the system address of the next contiguous data position. It can be accessed only if no transaction is executing (i.e. after a transaction has stopped). Read operations during transfer return an invalid value. The Host Driver (HD) shall initialize this register before starting a DMA transaction. After DMA has stopped, the next system address of the next contiguous data position can be read from this register. The DMA transfer waits at every boundary specified by the Host DMA Buffer Size in the Block Size register. The Host Controller generates DMA Interrupt to request to update this register. The HD sets the next system address of the next data position to this register. When most upper byte of this register (003h) is written, the HC restart the DMA transfer. When restarting DMA by the resume command or by setting Continue Request in the Block Gap Control register, the HC shall start at the next contiguous address stored here in the System Address register
Block_Size_Block_Count	0x00000004	32	mixed	0x00000000	Block size register Block count register
Argument	0x00000008	32	rw	0x00000000	Argument register
Transfer_Mode_Command	0x0000000C	29	mixed	0x00000000	Transfer mode register Command register
Response0	0x00000010	32	ro	0x00000000	Response register
Response1	0x00000014	32	ro	0x00000000	Response register
Response2	0x00000018	32	ro	0x00000000	Response register
Response3	0x0000001C	32	ro	0x00000000	Response register

Register Name	Address	Width	Type	Reset Value	Description
Buffer_Data_Port	0x00000020	32	rw	0x00000000	Buffer data port register
Present_State	0x00000024	25	ro	0x01F20000	Present State register
Host_control_Power_control_Block_Gap_Control_Wakeup_control	0x00000028	32	mixed	0x00000000	Host control register Power control register Block gap control register Wake-up control register
Clock_Control_Timeout_control_Software_reset	0x0000002C	27	mixed	0x00000000	Clock Control register Timeout control register Software reset register
Normal_interrupt_status_Error_interrupt_status	0x00000030	30	mixed	0x00000000	Normal interrupt status register Error interrupt status register
Normal_interrupt_status_enable_Error_interrupt_status_enable	0x00000034	30	mixed	0x00000000	Normal interrupt status enable register Error interrupt status enable register
Normal_interrupt_signal_enable_Error_interrupt_signal_enable	0x00000038	30	mixed	0x00000000	Normal interrupt signal enable register Error interrupt signal enable register
Auto_CMD12_error_status	0x0000003C	8	ro	0x00000000	Auto CMD12 error status register
Capabilities	0x00000040	31	ro	0x69EC0080	Capabilities register
Maximum_current_capabilities	0x00000048	24	ro	0x00000001	Maximum current capabilities register
Force_event_for_Auto_Cmd12_Error_Status_Force_event_register_for_error_interrupt_status	0x00000050	32	mixed	0x00000000	Force event register for Auto CMD12 error status register Force event register for error interrupt status
ADMA_error_status	0x00000054	3	mixed	0x00000000	ADMA error status register
ADMA_system_addresses	0x00000058	32	rw	0x00000000	ADMA system address register
Boot_Timeout_control	0x00000060	32	rw	0x00000000	Boot Timeout control register
Debug_Selection	0x00000064	1	wo	0x00000000	Debug Selection Register
SPI_interrupt_support	0x000000F0	8	rw	0x00000000	SPI interrupt support register
Slot_interrupt_status_Host_controller_version	0x000000FC	32	ro	0x89010000	Slot interrupt status register and Host controller version register

Register ([sdio](#)) SDMA_system_address_register

Name	SDMA_system_address_register
Relative Address	0x00000000
Absolute Address	sd0: 0xE0100000 sd1: 0xE0101000
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	This register contains the system memory address for a DMA transfer. When the Host Controller (HC) stops a DMA transfer, this register shall point to the system address of the next contiguous data position. It can be accessed only if no transaction is executing (i.e. after a transaction has stopped). Read operations during transfer return an invalid value. The Host Driver (HD) shall initialize this register before starting a DMA transaction. After DMA has stopped, the next system address of the next contiguous data position can be read from this register. The DMA transfer waits at every boundary specified by the Host DMA Buffer Size in the Block Size register. The Host Controller generates DMA Interrupt to request to update this register. The HD sets the next system address of the next data position to this register. When most upper byte of this register (003h) is written, the HC restart the DMA transfer. When restarting DMA by the resume command or by setting Continue Request in the Block Gap Control register, the HC shall start at the next contiguous address stored here in the System Address register

Register SDMA_system_address_register Details

Field Name	Bits	Type	Reset Value	Description
SDMA_System_Address	31:0	rw	0x0	Watchdog enable - if set, the watchdog is enabled and can generate any signals that are enabled.

Register ([sdio](#)) Block_Size_Block_Count

Name	Block_Size_Block_Count
Relative Address	0x00000004
Absolute Address	sd0: 0xE0100004 sd1: 0xE0101004
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Block size register Block count register

Register Block_Size_Block_Count Details

Field Name	Bits	Type	Reset Value	Description
Blocks_Count_for_Current_Transfer	31:16	rw	0x0	<p>This register is enabled when Block Count Enable in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. The HC decrements the block count after each block transfer and stops when the count reaches zero. It can be accessed only if no transaction is executing (i.e. after a transaction has stopped). Read operations during transfer return an invalid value and write operations shall be ignored. When saving transfer context as a result of Suspend command, the number of blocks yet to be transferred can be determined by reading this register. When restoring transfer context prior to issuing a Resume command, the HD shall restore the previously save block count.</p> <p>0000h - Stop Count 0001h - 1 block 0002h - 2 blocks --- --- FFFFh - 65535 blocks</p>
reserved	15	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
Host_SDMA_Buffer_Size	14:12	rw	0x0	Counter clock prescale - selects the prescaler division ratio: 00 = pclk divided by 8 01 = pclk divided by 64 10 = pclk divided by 256 11 = pclk divided by 4096 Note: If a restart signal is received the prescaler should be reset.
Transfer_Block_Size	11:0	rw	0x0	This register specifies the block size for block data transfers for CMD17, CMD18, CMD24, CMD25, and CMD53. It can be accessed only if no transaction is executing (i.e. after a transaction has stopped). Read operations during transfer return an invalid value and write operations shall be ignored. 0000h - No Data Transfer 0001h - 1 Byte 0002h - 2 Bytes 0003h - 3 Bytes 0004h - 4 Bytes --- --- 01FFh - 511 Bytes 0200h - 512 Bytes --- --- 0800h - 2048 Bytes

Register ([sdio](#)) Argument

Name	Argument
Relative Address	0x00000008
Absolute Address	sd0: 0xE0100008 sd1: 0xE0101008
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Argument register

Register Argument Details

Field Name	Bits	Type	Reset Value	Description
Command_Argument	31:0	rw	0x0	The SD Command Argument is specified as bit 39-8 of Command-Format.

Register ([sdio](#)) Transfer_Mode_Command

Name	Transfer_Mode_Command
Relative Address	0x0000000C
Absolute Address	sd0: 0xE010000C sd1: 0xE010100C
Width	29 bits
Access Type	mixed
Reset Value	0x00000000
Description	Transfer mode register Command register

Register Transfer_Mode_Command Details

Field Name	Bits	Type	Reset Value	Description
Command_Index	28:24	rw	0x0	This bit shall be set to the command number (CMD0-63, ACMD0-63).
Command_Type	23:22	rw	0x0	<p>There are three types of special commands. Suspend, Resume and Abort. These bits shall be set to 00b for all other commands. Suspend Command If the Suspend command succeeds, the HC shall assume the SD Bus has been released and that it is possible to issue the next command which uses the DAT line. The HC shall de-assert Read Wait for read transactions and stop checking busy for write transactions. The Interrupt cycle shall start, in 4-bit mode. If the Suspend command fails, the HC shall maintain its current state. and the HD shall restart the transfer by setting Continue Request in the Block Gap Control Register. Resume Command The HD re-starts the data transfer by restoring the registers in the range of 000-00Dh. The HC shall check for busy before starting write transfers. Abort Command If this command is set when executing a read transfer, the HC shall stop reads to the buffer. If this command is set when executing a write transfer, the HC shall stop driving the DAT line. After issuing the Abort command, the HD should issue a software reset</p> <p>00b - Normal 01b - Suspend 10b - Resume 11b - Abort</p>

Field Name	Bits	Type	Reset Value	Description
Data_Present_Select	21	rw	0x0	This bit is set to 1 to indicate that data is present and shall be transferred using the DAT line. If is set to 0 for the following: 1. Commands using only CMD line (ex. CMD52) 2. Commands with no data transfer but using busy signal on DAT[0] line (R1b or R5b ex. CMD38) 3. Resume Command 0 - No Data Present 1 - Data Present
Command_Index_Check_Enable	20	rw	0x0	If this bit is set to 1, the HC shall check the index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked. 0 - Disable 1 - Enable
Command_CRC_Check_Enable	19	rw	0x0	If this bit is set to 1, the HC shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. 0 - Disable 1 - Enable
reserved	18	ro	0x0	Reserved
Response_Type_Select	17:16	rw	0x0	Response Type Select 00 - No Response 01 - Response length 136 10 - Response length 48 11 - Response length 48 check Busy after response
reserved	15:6	ro	0x0	Reserved
Multi_Single_Block_Select	5	rw	0x0	This bit enables multiple block DAT line data transfers. 0 - Single Block 1 - Multiple Block
Data_Transfer_Direction_Select	4	rw	0x0	This bit defines the direction of DAT line data transfers. 0 - Write (Host to Card) 1 - Read (Card to Host)
reserved	3	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
Auto_CMD12_Enable	2	rw	0x0	Multiple block transfers for memory require CMD12 to stop the transaction. When this bit is set to 1, the HC shall issue CMD12 automatically when last block transfer is completed. The HD shall not set this bit to issue commands that do not require CMD12 to stop data transfer. 0 - Disable 1 - Enable
Block_Count_Enable	1	rw	0x0	This bit is used to enable the Block count register, which is only relevant for multiple block transfers. When this bit is 0, the Block Count register is disabled, which is useful in executing an infinite transfer. 0 - Disable 1 - Enable
DMA_Enable	0	rw	0x0	DMA can be enabled only if DMA Support bit in the Capabilities register is set. If this bit is set to 1, a DMA operation shall begin when the HD writes to the upper byte of Command register (00Fh). 0 - Disable 1 - Enable

Register ([sdio](#)) Response0

Name	Response0
Relative Address	0x00000010
Absolute Address	sd0: 0xE0100010 sd1: 0xE0101010
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Response register

Note: This register is the first in an array of 4 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
Response0	0xe0100010
Response1	0xe0100014
Response2	0xe0100018
Response3	0xe010001c

Register Response0 to Response3 Details

Field Name	Bits	Type	Reset Value	Description
Command_Response	31:0	ro	0x0	command responses registers

Register ([sdio](#)) Buffer_Data_Port

Name	Buffer_Data_Port
Relative Address	0x00000020
Absolute Address	sd0: 0xE0100020 sd1: 0xE0101020
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Buffer data port register

Register Buffer_Data_Port Details

Field Name	Bits	Type	Reset Value	Description
Buffer_Data	31:0	rw	0x0	The Host Controller Buffer can be accessed through this 32-bit Data Port Register.

Register ([sdio](#)) Present_State

Name	Present_State
Relative Address	0x00000024
Absolute Address	sd0: 0xE0100024 sd1: 0xE0101024
Width	25 bits
Access Type	ro
Reset Value	0x01F20000
Description	Present State register

Register Present_State Details

Field Name	Bits	Type	Reset Value	Description
CMD_Line_Signal_Level	24	ro	0x1	This status is used to check CMD line level to recover from errors, and for debugging.
DAT_Bit3_Bit0_Line_Signal_Level	23:20	ro	0xF	This status is used to check DAT line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from DAT[0]. D23 - DAT[3] D22 - DAT[2] D21 - DAT[1] D20 - DAT[0]
Write_Protect_Switch_Pin_Level	19	ro	0x0	The Write Protect Switch is supported for memory and combo cards. This bit reflects the SDWP# pin. 0 - Write protected (SDWP# = 1) 1 - Write enabled (SDWP# = 0)
Card_Detect_Pin_Level	18	ro	0x0	This bit reflects the inverse value of the SD CD# pin. 0 - No Card present (SD CD# = 1) 1 - Card present (SD CD# = 0)
Card_State_Stable	17	ro	0x1	This bit is used for testing. If it is 0, the Card Detect Pin Level is not stable. If this bit is set to 1, it means the Card Detect Pin Level is stable. The Software Reset For All in the Software Reset Register shall not affect this bit. 0 - Reset of Debouncing 1 - No Card or Inserted
Card_Inserted	16	ro	0x0	This bit indicates whether a card has been inserted. Changing from 0 to 1 generates a Card Insertion interrupt in the Normal Interrupt Status register and changing from 1 to 0 generates a Card Removal Interrupt in the Normal Interrupt Status register. The Software Reset For All in the Software Reset register shall not affect this bit. If a Card is removed while its power is on and its clock is oscillating, the HC shall clear SD Bus Power in the Power Control register and SD Clock Enable in the Clock control register. In addition the HD should clear the HC by the Software Reset For All in Software register. The card detect is active regardless of the SD Bus Power. 0 - Reset or Debouncing or No Card 1 - Card Inserted
reserved	15:12	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
Buffer_Read_Enable	11	ro	0x0	<p>This status is used for non-DMA read transfers. This read only flag indicates that valid data exists in the host side buffer status. If this bit is 1, readable data exists in the buffer. A change of this bit from 1 to 0 occurs when all the block data is read from the buffer. A change of this bit from 0 to 1 occurs when all the block data is ready in the buffer and generates the Buffer Read Ready Interrupt.</p> <p>0 - Read Disable 1 - Read Enable.</p>
Buffer_Write_Enable	10	ro	0x0	<p>This status is used for non-DMA write transfers. This read only flag indicates if space is available for write data. If this bit is 1, data can be written to the buffer. A change of this bit from 1 to 0 occurs when all the block data is written to the buffer. A change of this bit from 0 to 1 occurs when top of block data can be written to the buffer and generates the Buffer Write Ready Interrupt.</p> <p>0 - Write Disable 1 - Write Enable.</p>
Read_Transfer_Active	9	ro	0x0	<p>This status is used for detecting completion of a read transfer.</p> <p>This bit is set to 1 for either of the following conditions:</p> <ol style="list-style-type: none"> 1. After the end bit of the read command 2. When writing a 1 to continue Request in the Block Gap Control register to restart a read transfer <p>This bit is cleared to 0 for either of the following conditions:</p> <ol style="list-style-type: none"> 1. When the last data block as specified by block length is transferred to the system. 2. When all valid data blocks have been transferred to the system and no current block transfers are being sent as a result of the Stop At Block Gap Request set to 1. A transfer complete interrupt is generated when this bit changes to 0. <p>1 - Transferring data 0 - No valid data</p>

Field Name	Bits	Type	Reset Value	Description
Write_Transfer_Active	8	ro	0x0	<p>This status indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the HC. This bit is set in either of the following cases:</p> <ol style="list-style-type: none"> 1. After the end bit of the write command. 2. When writing a 1 to Continue Request in the Block Gap Control register to restart a write transfer. <p>This bit is cleared in either of the following cases:</p> <ol style="list-style-type: none"> 1. After getting the CRC status of the last data block as specified by the transfer count (Single or Multiple) 2. After getting a CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request. <p>During a write transaction, a Block Gap Event interrupt is generated when this bit is changed to 0, as a result of the Stop At Block Gap Request being set. This status is useful for the HD in determining when to issue commands during write busy.</p> <p>1 - transferring data 0 - No valid data</p>
reserved	7:3	ro	0x0	Reserved
DAT_Line_Active	2	ro	0x0	<p>This bit indicates whether one of the DAT line on SD bus is in use.</p> <p>1 - DAT line active 0 - DAT line inactive</p>

Field Name	Bits	Type	Reset Value	Description
Command_Inhibit_DAT	1	ro	0x0	<p>This status bit is generated if either the DAT Line Active or the Read transfer Active is set to 1. If this bit is 0, it indicates the HC can issue the next SD command. Commands with busy signal belong to Command Inhibit (DAT) (ex. R1b, R5b type). Changing from 1 to 0 generates a Transfer Complete interrupt in the Normal interrupt status register.</p> <p>Note: The SD Host Driver can save registers in the range of 000-00Dh for a suspend transaction after this bit has changed from 1 to 0.</p> <p>1 - cannot issue command which uses the DAT line 0 - Can issue command which uses the DAT line</p>
Command_Inhibit_CMD	0	ro	0x0	<p>If this bit is 0, it indicates the CMD line is not in use and the HC can issue a SD command using the CMD line. This bit is set immediately after the Command register (00Fh) is written. This bit is cleared when the command response is received. Even if the Command Inhibit (DAT) is set to 1, Commands using only the CMD line can be issued if this bit is 0. Changing from 1 to 0 generates a Command complete interrupt in the Normal Interrupt Status register. If the HC cannot issue the command because of a command conflict error or because of Command Not Issued By Auto CMD12 Error, this bit shall remain 1 and the Command Complete is not set. Status issuing Auto CMD12 is not read from this bit. Note: The SD host controller requires couple of clocks to update this register bit after the command is posted to command register.</p>

Register ([sdio](#))

Host_control_Power_control_Block_Gap_Control_Wakeup_control

Name	Host_control_Power_control_Block_Gap_Control_Wakeup_control
Relative Address	0x00000028
Absolute Address	sd0: 0xE0100028 sd1: 0xE0101028
Width	32 bits
Access Type	mixed
Reset Value	0x00000000

Description

Host control register

Power control register

Block gap control register

Wake-up control register

Register Host_control_Power_control_Block_Gap_Control_Wakeup_control Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:27	ro	0x0	Reserved
Wakeup_Event_Enable_On_SD_Card_Removal	26	rw	0x0	This bit enables wakeup event via Card Removal assertion in the Normal Interrupt Status register. FN_WUS (Wake up Support) in CIS does not affect this bit. 1 - Enable 0 - Disable
Wakeup_Event_Enable_On_SD_Card_Insertion	25	rw	0x0	This bit enables wakeup event via Card Insertion assertion in the Normal Interrupt Status register. FN_WUS (Wake up Support) in CIS does not affect this bit. 1 - Enable 0 - Disable
Wakeup_Event_Enable_On_Card_Interrupt	24	rw	0x0	This bit enables wakeup event via Card Interrupt assertion in the Normal Interrupt Status register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. 1 - Enable 0 - Disable
reserved	23:20	ro	0x0	Reserved
Interrupt_At_Block_Gap	19	rw	0x0	This bit is valid only in 4-bit mode of the SDIO card and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. If the SD card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0. When the HD detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card.

Field Name	Bits	Type	Reset Value	Description
Read_Wait_Control	18	rw	0x0	<p>The read wait function is optional for SDIO cards. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using DAT[2] line. Otherwise the HC has to stop the SD clock to hold read data, which restricts commands generation. When the HD detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card. If the card does not support read wait, this bit shall never be set to 1 otherwise DAT line conflict may occur. If this bit is set to 0, Suspend / Resume cannot be supported</p> <p>1 - Enable Read Wait Control 0 - Disable Read Wait Control</p>
Continue_Request	17	rw	0x0	<p>This bit is used to restart a transaction which was stopped using the Stop At Block Gap Request. To cancel stop at the block gap, set Stop At block Gap Request to 0 and set this bit to restart the transfer.</p> <p>The HC automatically clears this bit in either of the following cases:</p> <p>1) In the case of a read transaction, the DAT Line Active changes from 0 to 1 as a read transaction restarts.</p> <p>2) In the case of a write transaction, the Write transfer active changes from 0 to 1 as the write transaction restarts.</p> <p>Therefore it is not necessary for Host driver to set this bit to 0. If Stop At Block Gap Request is set to 1, any write to this bit is ignored.</p> <p>1 - Restart 0 - Ignored</p>

Field Name	Bits	Type	Reset Value	Description
Stop_At_Block_Gap_Request	16	rw	0x0	<p>This bit is used to stop executing a transaction at the next block gap for non- DMA,SDMA and ADMA transfers. Until the transfer complete is set to 1, indicating a transfer completion the HD shall leave this bit set to 1. Clearing both the Stop At Block Gap Request and Continue Request shall not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The HC shall honor Stop At Block Gap Request for write transfers, but for read transfers it requires that the SD card support Read Wait. Therefore the HD shall not set this bit during read transfers unless the SD card supports Read Wait and has set Read Wait Control to 1. In case of write transfers in which the HD writes data to the Buffer Data Port register, the HD shall set this bit after all block data is written. If this bit is set to 1, the HD shall not write data to Buffer data port register. This bit affects Read Transfer Active, Write Transfer Active, DAT line active and Command Inhibit (DAT) in the Present State register.</p> <p>1 - Stop 0 - Transfer</p>
reserved	15:12	ro	0x0	Reserved
SD_Bus_Voltage_Select	11:9	rw	0x0	<p>By setting these bits, the HD selects the voltage level for the SD card. Before setting this register, the HD shall check the voltage support bits in the capabilities register. If an unsupported voltage is selected, the</p> <p>Host System shall not supply SD bus voltage</p> <p>111b - 3.3 Flattop.) 110b - 3.0 V(Typ.) 101b - 1.8 V(Typ.) 100b - 000b - Reserved</p>
SD_Bus_Power	8	rw	0x0	<p>Before setting this bit, the SD host driver shall set SD Bus Voltage Select. If the HC detects the No Card State, this bit shall be cleared.</p> <p>1 - Power on 0 - Power off</p>
Card_detect_signal_detection	7	rw	0x0	<p>This bit selects source for card detection.</p> <p>1- The card detect test level is selected 0 -SDCD# is selected (for normal use)</p>

Field Name	Bits	Type	Reset Value	Description
Card_Detect_Test_Level	6	rw	0x0	This bit is enabled while the Card Detect Signal Selection is set to 1 and it indicates card inserted or not. Generates (card ins or card removal) interrupt when the normal int sts enable bit is set. 1 - Card Inserted 0 - No Card
reserved	5	ro	0x0	Reserved
DMA_Select	4:3	rw	0x0	One of supported DMA modes can be selected. The host driver shall check support of DMA modes by referring the Capabilities register. 00 - SDMA is selected 01 - 32-bit Address ADMA1 is selected 10 -32-bit Address ADMA2 is selected 11 - 64-bit Address ADMA2 is selected
High_Speed_Enable	2	rw	0x0	This bit is optional. Before setting this bit, the HD shall check the High Speed Support in the capabilities register. If this bit is set to 0 (default), the HC outputs CMD line and DAT lines at the falling edge of the SD clock (up to 25 MHz/20 MHz for MMC). If this bit is set to 1, the HC outputs CMD line and DAT lines at the rising edge of the SD clock (up to 50 MHz for SD/52 MHz for MMC) 1 - High Speed Mode 0 - Normal Speed Mode
Data_Transfer_Width_SD1_or_SD4	1	rw	0x0	This bit selects the data width of the HC. The HD shall select it to match the data width of the SD card. 1 - 4 bit mode 0 - 1 bit mode
LED_Control	0	rw	0x0	This bit is used to caution the user not to remove the card while the SD card is being accessed. If the software is going to issue multiple SD commands, this bit can be set during all transactions. It is not necessary to change for each transaction. 1 - LED on 0 - LED off

Register ([sdio](#)) Clock_Control_Timeout_control_Software_reset

Name Clock_Control_Timeout_control_Software_reset
Relative Address 0x0000002C

Absolute Address	sd0: 0xE010002C sd1: 0xE010102C
Width	27 bits
Access Type	mixed
Reset Value	0x00000000
Description	Clock Control register Timeout control register Software reset register

Register Clock_Control_Timeout_control_Software_reset Details

Field Name	Bits	Type	Reset Value	Description
Software_Reset_for_D AT_Line	26	rw	0x0	Only part of data circuit is reset. The following registers and bits are cleared by this bit: Buffer Data Port Register Buffer is cleared and Initialized. Present State register Buffer read Enable Buffer write Enable Read Transfer Active Write Transfer Active DAT Line Active Command Inhibit (DAT) Block Gap Control register Continue Request Stop At Block Gap Request Normal Interrupt Status register Buffer Read Ready Buffer Write Ready Block Gap Event Transfer Complete 1 - Reset 0 - Work
Software_Reset_for_C MD_Line	25	rw	0x0	Only part of command circuit is reset. The following registers and bits are cleared by this bit: Present State register Command Inhibit (CMD) Normal Interrupt Status register Command Complete 1 - Reset 0 - Work

Field Name	Bits	Type	Reset Value	Description
Software_Reset_for_All	24	rw	0x0	<p>This reset affects the entire HC except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared to 0. During its initialization, the HD shall set this bit to 1 to reset the HC. The HC shall reset this bit to 0 when capabilities registers are valid and the HD can read them. Additional use of Software Reset For All may not affect the value of the Capabilities registers. If this bit is set to 1, the SD card shall reset itself and must be re initialized by the HD.</p> <p>1 - Reset 0 - Work</p>
reserved	23:20	ro	0x0	Reserved
Data_Timeout_Counter_Value_	19:16	rw	0x0	<p>This value determines the interval by which DAT line time-outs are detected. Refer to the Data Timeout Error in the Error Interrupt Status register for information on factors that dictate Timeout generation. Timeout clock frequency will be generated by dividing the sdclockTMCLK by this value. When setting this register, prevent inadvertent Timeout events by clearing the Data Time-out Error Status Enable (in the Error Interrupt Status Enable register)</p> <p>1111 - Reserved 1110 - $TMCLK * 2^{27}$ ----- ----- 0001 - $TMCLK * 2^{14}$ 0000 - $TMCLK * 2^{13}$</p>

Field Name	Bits	Type	Reset Value	Description
SDCLK_Frequency_Select	15:8	rw	0x0	<p>This register is used to select the frequency of the SDCLK pin. The frequency is not programmed directly; rather this register holds the divisor of the Base Clock Frequency For SD clock in the capabilities register. Only the following settings are allowed.</p> <p>80h - base clock divided by 256 40h - base clock divided by 128 20h - base clock divided by 64 10h - base clock divided by 32 08h - base clock divided by 16 04h - base clock divided by 8 02h - base clock divided by 4 01h - base clock divided by 2 00h - base clock(10MHz-63MHz)</p> <p>Setting 00h specifies the highest frequency of the SD Clock. When setting multiple bits, the most significant bit is used as the divisor. But multiple bits should not be set. The two default divider values can be calculated by the frequency that is defined by the Base Clock Frequency For SD Clock in the Capabilities register.</p> <p>1) 25 MHz divider value 2) 400 KHz divider value</p> <p>The frequency of the SDCLK is set by the following formula:</p> <p>$\text{Clock Frequency} = (\text{Baseclock}) / \text{divisor}$.</p> <p>Thus choose the smallest possible divisor which results in a clock frequency that is less than or equal to the target frequency.</p> <p>Maximum Frequency for SD = 50Mhz (base clock) Maximum Frequency for MMC = 52Mhz (base clock) Minimum Frequency = 195.3125Khz (50Mhz / 256), same calc for MMC also</p>
reserved	7:3	ro	0x0	Reserved
SD_Clock_Enable	2	rw	0x0	<p>The HC shall stop SDCLK when writing this bit to 0. SDCLK frequency Select can be changed when this bit is 0. Then, the HC shall maintain the same clock frequency until SDCLK is stopped (Stop at SDCLK = 0). If the HC detects the No Card state, this bit shall be cleared.</p> <p>1 - Enable 0 - Disable</p>

Field Name	Bits	Type	Reset Value	Description
Internal_Clock_Stable	1	ro	0x0	This bit is set to 1 when SD clock is stable after writing to Internal Clock Enable in this register to 1. The SD Host Driver shall wait to set SD Clock Enable until this bit is set to 1. Note: This is useful when using PLL for a clock oscillator that requires setup time. 1 - Ready 0 - Not Ready
Internal_Clock_Enable	0	rw	0x0	This bit is set to 0 when the HD is not using the HC or the HC awaits a wakeup event. The HC should stop its internal clock to go very low power state. Still, registers shall be able to be read and written. Clock starts to oscillate when this bit is set to 1. When clock oscillation is stable, the HC shall set Internal Clock Stable in this register to 1. This bit shall not affect card detection. 1 - Oscillate 0 - Stop

Register ([sdio](#)) Normal_interrupt_status_Error_interrupt_status

Name	Normal_interrupt_status_Error_interrupt_status
Relative Address	0x00000030
Absolute Address	sd0: 0xE0100030 sd1: 0xE0101030
Width	30 bits
Access Type	mixed
Reset Value	0x00000000
Description	Normal interrupt status register Error interrupt status register

Register Normal_interrupt_status_Error_interrupt_status Details

Field Name	Bits	Type	Reset Value	Description
Ceata_Error_Status	29	wtc	0x0	Occurs when ATA command termination has occurred due to an error condition the device has encountered. 0 - no error 1 - error
Target_Response_error	28	wtc	0x0	Occurs when detecting ERROR in m_hresp(dma transaction) 0 - no error 1 - error

Field Name	Bits	Type	Reset Value	Description
reserved	27:26	ro	0x0	Reserved
ADMA_Error	25	wtc	0x0	This bit is set when the Host Controller detects errors during ADMA based data transfer. The state of the ADMA at an error occurrence is saved in the ADMA Error Status Register. 1- Error 0 -No error
Auto_CMD12_Error	24	wtc	0x0	Occurs when detecting that one of the bits in Auto CMD12 Error Status register has changed from 0 to 1. This bit is set to 1 also when Auto CMD12 is not executed due to the previous command error. 0 - No Error 1 - Error
Current_Limit_Error	23	wtc	0x0	By setting the SD Bus Power bit in the Power Control Register, the HC is requested to supply power for the SD Bus. If the HC supports the Current Limit Function, it can be protected from an Illegal card by stopping power supply to the card in which case this bit indicates a failure status. Reading 1 means the HC is not supplying power to SD card due to some failure. Reading 0 means that the HC is supplying power and no error has occurred. This bit shall always set to be 0, if the HC does not support this function. 0 - No Error 1 - Power Fail
Data_End_Bit_Error	22	wtc	0x0	Occurs when detecting 0 at the end bit position of read data which uses the DAT line or the end bit position of the CRC status. 0 - No Error 1 - Error
Data_CRC_Error	21	wtc	0x0	Occurs when detecting CRC error when transferring read data which uses the DAT line or when detecting the Write CRC Status having a value of other than '010'. 0 - No Error 1 - Error
Data_Timeout_Error	20	wtc	0x0	Occurs when detecting one of following timeout conditions. 1. Busy Timeout for R1b, R5b type. 2. Busy Timeout after Write CRC status 3. Write CRC status Timeout 4. Read Data Timeout 0 - No Error 1 - Timeout

Field Name	Bits	Type	Reset Value	Description
Command_Index_Error	19	wtc	0x0	Occurs if a Command Index error occurs in the Command Response. 0 - No Error 1 - Error
Command_End_Bit_Error	18	wtc	0x0	Occurs when detecting that the end bit of a command response is 0. 0 - No Error 1 - End Bit Error Generated
Command_CRC_Error	17	wtc	0x0	Command CRC Error is generated in two cases. 1. If a response is returned and the Command Timeout Error is set to 0, this bit is set to 1 when detecting a CRT error in the command response 2. The HC detects a CMD line conflict by monitoring the CMD line when a command is issued. If the HC drives the CMD line to 1 level, but detects 0 level on the CMD line at the next SDCLK edge, then the HC shall abort the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error shall also be set to 1 to distinguish CMD line conflict. 0 - No Error 1 - CRC Error Generated
Command_Timeout_Error	16	wtc	0x0	Occurs only if the no response is returned within 64 SDCLK cycles from the end bit of the command. If the HC detects a CMD line conflict, in which case Command CRC Error shall also be set. This bit shall be set without waiting for 64 SDCLK cycles because the command will be aborted by the HC. 0 - No Error 1 - Timeout
Error_Interrupt	15	ro	0x0	If any of the bits in the Error Interrupt Status Register are set, then this bit is set. Therefore the HD can test for an error by checking this bit first. 0 - No Error. 1 - Error.
reserved	14:11	ro	0x0	Reserved
Boot_terminate Interrupt	10	wtc	0x0	This status is set if the boot operation get terminated 0 - Boot operation is not terminated. 1 - Boot operation is terminated
Boot_ack_rcv	9	wtc	0x0	This status is set if the boot acknowledge is received from device. 0 - Boot ack is not received. 1 - Boot ack is received.

Field Name	Bits	Type	Reset Value	Description
Card_Interrupt	8	ro	0x0	<p>Writing this bit to 1 does not clear this bit. It is cleared by resetting the SD card interrupt factor. In 1-bit mode, the HC shall detect the Card Interrupt without SD Clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so there are some sample delays between the interrupt signal from the card and the interrupt to the Host system.</p> <p>when this status has been set and the HD needs to start this interrupt service, Card Interrupt Status Enable in the Normal Interrupt Status register shall be set to 0 in order to clear the card interrupt statuses latched in the HC and stop driving the Host System. After completion of the card interrupt service (the reset factor in the SD card and the interrupt signal may not be asserted), set Card Interrupt Status Enable to 1 and start sampling the interrupt signal again.</p> <p>0 - No Card Interrupt 1 - Generate Card Interrupt</p>
Card_Removal	7	wtc	0x0	<p>This status is set if the Card Inserted in the Present State register changes from 1 to 0. When the HD writes this bit to 1 to clear this status the status of the Card Inserted in the Present State register should be confirmed.</p> <p>Because the card detect may possibly be changed when the HD clear this bit an Interrupt event may not be generated.</p> <p>0 - Card State Stable or Debouncing 1 - Card Removed</p>
Card_Insertion	6	wtc	0x0	<p>This status is set if the Card Inserted in the Present State register changes from 0 to 1. When the HD writes this bit to 1 to clear this status the status of the Card Inserted in the Present State register should be confirmed.</p> <p>Because the card detect may possibly be changed when the HD clear this bit an Interrupt event may not be generated.</p> <p>0 - Card State Stable or Debouncing 1 - Card Inserted</p>
Buffer_Read_Ready	5	wtc	0x0	<p>This status is set if the Buffer Read Enable changes from 0 to 1.</p> <p>0 - Not Ready to read Buffer. 1 - Ready to read Buffer.</p>

Field Name	Bits	Type	Reset Value	Description
Buffer_Write_Ready	4	wtc	0x0	This status is set if the Buffer Write Enable changes from 0 to 1. 0 - Not Ready to Write Buffer. 1 - Ready to Write Buffer.
DMA_Interrupt	3	wtc	0x0	This status is set if the HC detects the Host DMA Buffer Boundary in the Block Size register. 0 - No DMA Interrupt 1 - DMA Interrupt is Generated
Block_Gap_Event	2	wtc	0x0	If the Stop At Block Gap Request in the Block Gap Control Register is set, this bit is set. Read Transaction: This bit is set at the falling edge of the DAT Line Active Status (When the transaction is stopped at SD Bus timing. The Read Wait must be supported in order to use this function). Write Transaction: This bit is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing). 0 - No Block Gap Event 1 - Transaction stopped at Block Gap

Field Name	Bits	Type	Reset Value	Description
Transfer_Complete	1	wtc	0x0	<p>This bit is set when a read / write transaction is completed.</p> <p>Read Transaction:</p> <p>This bit is set at the falling edge of Read Transfer Active Status.</p> <p>There are two cases in which the Interrupt is generated. The first is when a data transfer is completed as specified by data length (After the last data has been read to the Host System). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request in the Block Gap Control Register (After valid data has been read to the Host System).</p> <p>Write Transaction:</p> <p>This bit is set at the falling edge of the DAT Line Active Status.</p> <p>There are two cases in which the Interrupt is generated. The first is when the last data is written to the card as specified by data length and Busy signal is released. The second is when data transfers are stopped at the block gap by setting Stop At Block Gap Request in the Block Gap Control Register and data transfers completed. (After valid data is written to the SD card and the busy signal is released).</p> <p>Note: Transfer Complete has higher priority than Data Timeout Error. If both bits are set to 1, the data transfer can be considered complete</p> <p>0 - No Data Transfer Complete 1 - Data Transfer Complete</p>
Command_Complete	0	wtc	0x0	<p>This bit is set when get the end bit of the command response (Except Auto CMD12).</p> <p>Note: Command Timeout Error has higher priority than Command Complete. If both are set to 1, it can be considered that the response was not received correctly.</p> <p>0 - No Command Complete 1 - Command Complete</p>

Register ([sdio](#))

Normal_interrupt_status_enable_Error_interrupt_status_enable

Name	Normal_interrupt_status_enable_Error_interrupt_status_enable
Relative Address	0x00000034

Absolute Address	sd0: 0xE0100034 sd1: 0xE0101034
Width	30 bits
Access Type	mixed
Reset Value	0x00000000
Description	Normal interrupt status enable register Error interrupt status enable register

Register Normal_interrupt_status_enable_Error_interrupt_status_enable Details

Field Name	Bits	Type	Reset Value	Description
Ceata_Error_Status_Enable	29	rw	0x0	0 - Masked 1 - Enabled
Target_Response_Error_Status_Enable	28	rw	0x0	0 - Masked 1 - Enabled
reserved	27:26	ro	0x0	Reserved
ADMA_Error_Status_Enable	25	rw	0x0	0 - Masked 1 - Enabled
Auto_CMD12_Error_Status_Enable	24	rw	0x0	0 - Masked 1 - Enabled
Current_Limit_Error_Status_Enable	23	rw	0x0	0 - Masked 1 - Enabled
Data_End_Bit_Error_Status_Enable	22	rw	0x0	0 - Masked 1 - Enabled
Data_CRC_Error_Status_Enable	21	rw	0x0	0 - Masked 1 - Enabled
Data_Timeout_Error_Status_Enable	20	rw	0x0	0 - Masked 1 - Enabled
Command_Index_Error_Status_Enable	19	rw	0x0	0 - Masked 1 - Enabled
Command_End_Bit_Error_Status_Enable	18	rw	0x0	0 - Masked 1 - Enabled
Command_CRC_Error_Status_Enable	17	rw	0x0	0 - Masked 1 - Enabled
Command_Timeout_Error_Status_Enable	16	rw	0x0	0 - Masked 1 - Enabled
Fixed_to_0	15	ro	0x0	The HC shall control error Interrupts using the Error Interrupt Status Enable register.
reserved	14:11	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
Boot_terminate_Interru pt_enable	10	rw	0x0	0 - Masked 1 - Enabled
Boot_ack_rcv_enable	9	rw	0x0	0 - Masked 1 - Enabled
Card_Interrupt_Status _Enable	8	rw	0x0	If this bit is set to 0, the HC shall clear Interrupt request to the System. The Card Interrupt detection is stopped when this bit is cleared and restarted when this bit is set to 1. The HD should clear the Card Interrupt Status Enable before servicing the Card Interrupt and should set this bit again after all Interrupt requests from the card are cleared to prevent inadvertent Interrupts. 0 - Masked 1 - Enabled
Card_Removal_Status_ Enable	7	rw	0x0	0 - Masked 1 - Enabled
Card_Insertion_Status_ Enable	6	rw	0x0	0 - Masked 1 - Enabled
Buffer_Read_Ready_St atus_Enable	5	rw	0x0	0 - Masked 1 - Enabled
Buffer_Write_Ready_St atus_Enable	4	rw	0x0	0 - Masked 1 - Enabled
DMA_Interrupt_Status _Enable	3	rw	0x0	0 - Masked 1 - Enabled
Block_Gap_Event_Stat us_Enable	2	rw	0x0	0 - Masked 1 - Enabled
Transfer_Complete_Sta tus_Enable	1	rw	0x0	0 - Masked 1 - Enabled
Command_Complete_ Status_Enable	0	rw	0x0	0 - Masked 1 - Enabled

Register ([sdio](#))

Normal_interrupt_signal_enable_Error_interrupt_signal_enable

Name	Normal_interrupt_signal_enable_Error_interrupt_signal_enable
Relative Address	0x00000038
Absolute Address	sd0: 0xE0100038 sd1: 0xE0101038
Width	30 bits
Access Type	mixed

Reset Value 0x00000000

Description Normal interrupt signal enable register
Error interrupt signal enable register

Register Normal_interrupt_signal_enable_Error_interrupt_signal_enable Details

Field Name	Bits	Type	Reset Value	Description
Ceata_Error_Signal_Enable	29	rw	0x0	0 - Masked 1 - Enabled
Target_Response_Error_Signal_Enable	28	rw	0x0	0 - Masked 1 - Enabled
reserved	27:26	ro	0x0	Reserved
ADMA_Error_Signal_Enable	25	rw	0x0	0 - Masked 1 - Enabled
Auto_CMD12_Error_Signal_Enable	24	rw	0x0	0 - Masked 1 - Enabled
Current_Limit_Error_Signal_Enable	23	rw	0x0	0 - Masked 1 - Enabled
Data_End_Bit_Error_Signal_Enable	22	rw	0x0	0 - Masked 1 - Enabled
Data_CRC_Error_Signal_Enable	21	rw	0x0	0 - Masked 1 - Enabled
Data_Timeout_Error_Signal_Enable	20	rw	0x0	0 - Masked 1 - Enabled
Command_Index_Error_Signal_Enable	19	rw	0x0	0 - Masked 1 - Enabled
Command_End_Bit_Error_Signal_Enable	18	rw	0x0	0 - Masked 1 - Enabled
Command_CRC_Error_Signal_Enable	17	rw	0x0	0 - Masked 1 - Enabled
Command_Timeout_Error_Signal_Enable	16	rw	0x0	0 - Masked 1 - Enabled
Fixed_to_0	15	ro	0x0	The HD shall control error Interrupts using the Error Interrupt Signal Enable register.
reserved	14:11	ro	0x0	Reserved
Boot_terminate_interrupt_signal_enable	10	rw	0x0	0 - Masked 1 - Enabled
Boot_ack_rcv_signal_enable	9	rw	0x0	0 - Masked 1 - Enabled

Field Name	Bits	Type	Reset Value	Description
Card_Interrupt_Signal_Enable	8	rw	0x0	0 - Masked 1 - Enabled
Card_Removal_Signal_Enable	7	rw	0x0	0 - Masked 1 - Enabled
Card_Insertion_Signal_Enable	6	rw	0x0	0 - Masked 1 - Enabled
Buffer_Read_Ready_Signal_Enable	5	rw	0x0	0 - Masked 1 - Enabled
Buffer_Write_Ready_Signal_Enable	4	rw	0x0	0 - Masked 1 - Enabled
DMA_Interrupt_Signal_Enable	3	rw	0x0	0 - Masked 1 - Enabled
Block_Gap_Event_Signal_Enable	2	rw	0x0	0 - Masked 1 - Enabled
Transfer_Complete_Signal_Enable	1	rw	0x0	0 - Masked 1 - Enabled
Command_Complete_Signal_Enable	0	rw	0x0	0 - Masked 1 - Enabled

Register ([sdio](#)) Auto_CMD12_error_status

Name	Auto_CMD12_error_status
Relative Address	0x0000003C
Absolute Address	sd0: 0xE010003C sd1: 0xE010103C
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	Auto CMD12 error status register

Register Auto_CMD12_error_status Details

Field Name	Bits	Type	Reset Value	Description
Command_Not_Issued_By_Auto_CMD12_Error	7	ro	0x0	Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 error (D04 - D01) in this register. 0 - No Error 1 - Not Issued
reserved	6:5	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
Auto_CMD12_Index_Error	4	ro	0x0	Occurs if the Command Index error occurs in response to a command. 0 - No Error 1 - Error
Auto_CMD12_End_Bit_Error	3	ro	0x0	Occurs when detecting that the end bit of command response is 0. 0 - No Error 1 - End Bit Error Generated
Auto_CMD12_CRC_Error	2	ro	0x0	Occurs when detecting a CRC error in the command response. 0 - No Error 1 - CRC Error Generated
Auto_CMD12_Timeout_Error	1	ro	0x0	Occurs if the no response is returned within 64 SDCLK cycles from the end bit of the command. If this bit is set to 1, the other error status bits (D04 - D02) are meaningless. 0 - No Error 1 - Timeout
Auto_CMD12_not_Executed	0	ro	0x0	If memory multiple block data transfer is not started due to command error, this bit is not set because it is not necessary to issue Auto CMD12. Setting this bit to 1 means the HC cannot issue Auto CMD12 to stop memory multiple block transfer due to some error. If this bit is set to 1, other error status bits (D04 - D01) are meaningless. 0 - Executed 1 - Not Executed

Register ([sdio](#)) Capabilities

Name	Capabilities
Relative Address	0x00000040
Absolute Address	sd0: 0xE0100040 sd1: 0xE0101040
Width	31 bits
Access Type	ro
Reset Value	0x69EC0080
Description	Capabilities register

Register Capabilities Details

Field Name	Bits	Type	Reset Value	Description
Spi_block_mode	30	ro	0x1	Spi block mode 0 - Not Supported 1 - Supported
Spi_mode	29	ro	0x1	Spi mode 0 - Not Supported 1 - Supported
64_bit_System_Bus_Support	28	ro	0x0	1 - supports 64 bit system address 0 - Does not support 64 bit system address
Interrupt_mode	27	ro	0x1	Interrupt mode 0 - Not Supported 1 - Supported
Voltage_Support_1_8_V	26	ro	0x0	0 - 1.8 V Not Supported 1 - 1.8 V Supported
Voltage_Support_3_0_V	25	ro	0x0	0 - 3.0 V Not Supported 1 - 3.0 V Supported
Voltage_Support_3_3_V	24	ro	0x1	0 - 3.3 V Not Supported 1 - 3.3 V Supported
Suspend_Resume_Support	23	ro	0x1	This bit indicates whether the HC supports Suspend / Resume functionality. If this bit is 0, the Suspend and Resume mechanism are not supported and the HD shall not issue either Suspend / Resume commands. 0 - Not Supported 1 - Supported
SDMA_Support	22	ro	0x1	This bit indicates whether the HC is capable of using DMA to transfer data between system memory and the HC directly. 0 - SDMA Not Supported 1 - SDMA Supported.
High_Speed_Support	21	ro	0x1	This bit indicates whether the HC and the Host System support High Speed mode and they can supply SD Clock frequency from 25Mhz to 50 MHz (for SD)/ 20MHz to 52MHz (for MMC). 0 - High Speed Not Supported 1 - High Speed Supported
reserved	20	ro	0x0	Reserved
ADMA2_Support	19	ro	0x1	1 - ADMA2 support. 0 - ADMA2 not support

Field Name	Bits	Type	Reset Value	Description
Extended_Media_Bus_Support	18	ro	0x1	This bit indicates whether the Host Controller is capable bus. 1 - Extended Media Bus Supported 0 - Extended Media Bus not Supported
Max_Block_Length	17:16	ro	0x0	This value indicates the maximum block size that the HD can read and write to the buffer in the HC. The buffer shall transfer this block size without wait cycles. Three sizes can be defined as indicated below. 00 - 512 byte 01 - 1024 byte 10 - 2048 byte 11 - 4096 byte
reserved	15:14	ro	0x0	Reserved
reserved	13:8	ro	0x0	Reserved. Do not modify.
Timeout_Clock_Unit	7	ro	0x1	This bit shows the unit of base clock frequency used to detect Data Timeout Error. 0 - KHz 1 - MHz
reserved	6	ro	0x0	Reserved
reserved	5:0	ro	0x0	Reserved. Do not modify.

Register ([sdio](#)) Maximum_current_capabilities

Name	Maximum_current_capabilities
Relative Address	0x00000048
Absolute Address	sd0: 0xE0100048 sd1: 0xE0101048
Width	24 bits
Access Type	ro
Reset Value	0x00000001
Description	Maximum current capabilities register

Register Maximum_current_capabilities Details

Field Name	Bits	Type	Reset Value	Description
Maximum_Current_for_1_8V	23:16	ro	0x0	Maximum Current for 1.8V

Field Name	Bits	Type	Reset Value	Description
Maximum_Current_for_3_0V	15:8	ro	0x0	Maximum Current for 3.0V
Maximum_Current_for_3_3V	7:0	ro	0x1	Maximum Current for 3.3V

Register ([sdio](#))

Force_event_for_AutoCmd12_Error_Status_Force_event_register_for_error_interrupt_status

Name	Force_event_for_AutoCmd12_Error_Status_Force_event_register_for_error_interrupt_status
Relative Address	0x00000050
Absolute Address	sd0: 0xE0100050 sd1: 0xE0101050
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Force event register for Auto CMD12 error status register Force event register for error interrupt status

Register

Force_event_for_AutoCmd12_Error_Status_Force_event_register_for_error_interrupt_status Details

Field Name	Bits	Type	Reset Value	Description
Force_Event_for_Vendor_Specific_Error_Status	31:30	wo	0x0	Additional status bits can be defined in this register by the vendor. 1 - Interrupt is generated 0 - No interrupt
Force_Event_for_Ceata_error	29	wo	0x0	Force Event for Ceata Error 1 - Interrupt is generated 0 - No interrupt
Force_event_for_Target_Response_error	28	wo	0x0	Force Event for Target Response Error 1 - Interrupt is generated 0 - No interrupt
reserved	27:26	ro	0x0	Reserved
Force_Event_for_ADMA_Error	25	wo	0x0	Force Event for ADMA Error 1 - Interrupt is generated 0 - No interrupt

Field Name	Bits	Type	Reset Value	Description
Force_Event_for_Auto_CMD12_Error	24	wo	0x0	Force Event for Auto CMD12 Error 1 - Interrupt is generated 0 - No interrupt
Force_Event_for_Current_Limit_Error	23	wo	0x0	Force Event for Current Limit Error 1 - Interrupt is generated 0 - No interrupt
Force_Event_for_Data_End_Bit_Error	22	wo	0x0	Force Event for Data End Bit Error 1 - Interrupt is generated 0 - No interrupt
Force_Event_for_Data_CRC_Error	21	wo	0x0	Force Event for Data CRC Error 1 - Interrupt is generated 0 - No interrupt
Force_Event_for_Data_Timeout_Error	20	wo	0x0	Force Event for Data Timeout Error 1 - Interrupt is generated 0 - No interrupt
Force_Event_for_Command_Index_Error	19	wo	0x0	Force Event for Command Index Error 1 - Interrupt is generated 0 - No interrupt
Force_Event_for_Command_End_Bit_Error	18	wo	0x0	Force Event for Command End Bit Error 1 - Interrupt is generated 0 - No interrupt
Force_Event_for_Command_CRC_Error	17	wo	0x0	Force Event for Command CRC Error 1 - Interrupt is generated 0 - No interrupt
Force_Event_for_Command_Timeout_Error	16	wo	0x0	Force Event for Command Timeout Error 1 - Interrupt is generated 0 - No interrupt
reserved	15:8	ro	0x0	Reserved
Force_Event_for_command_not_issued_by_Auto_CMD12_Error	7	wo	0x0	1 - Interrupt is generated 0 - no interrupt
reserved	6:5	ro	0x0	Reserved
Force_Event_for_Auto_CMD12_Index_Error	4	wo	0x0	1 - Interrupt is generated 0 - no interrupt
Force_Event_for_Auto_CMD12_End_bit_Error	3	wo	0x0	1 - Interrupt is generated 0 - no interrupt
Force_Event_for_Auto_CMD12_CRC_Error	2	wo	0x0	1 - Interrupt is generated 0 - no interrupt

Field Name	Bits	Type	Reset Value	Description
Force_Event_for_Auto_CMD12_timeout_Error	1	wo	0x0	1 - Interrupt is generated 0 - no interrupt
Force_Event_for_Auto_CMD12_NOT_Executed	0	wo	0x0	1 - Interrupt is generated 0 - no interrupt

Register ([sdio](#)) ADMA_error_status

Name	ADMA_error_status
Relative Address	0x00000054
Absolute Address	sd0: 0xE0100054 sd1: 0xE0101054
Width	3 bits
Access Type	mixed
Reset Value	0x00000000
Description	ADMA error status register

Register ADMA_error_status Details

Field Name	Bits	Type	Reset Value	Description
ADMA_Length_Mismatch_Error	2	wtc	0x0	<p>This error occurs in the following 2 cases.</p> <ol style="list-style-type: none"> 1. While Block Count Enable being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length. 2. Total data length can not be divided by the block length. <p>1 - Error 0 - No error</p>
ADMA_Error_State	1:0	ro	0x0	<p>This field indicates the state of ADMA when error is occurred during ADMA data transfer. This field never indicates "10" because ADMA never stops in this state.</p> <p>D01 - D00 : ADMA Error State when error is occurred</p> <p>Contents of SYS_SDR register</p> <p>00 - ST_STOP (Stop DMA) Points next of the error descriptor</p> <p>01 - ST_FDS (Fetch Descriptor) Points the error descriptor</p> <p>10 - Never set this state (Not used)</p> <p>11 - ST_TFR (Transfer Data) Points the next of the error descriptor</p>

Register ([sdio](#)) ADMA_system_address

Name	ADMA_system_address
Relative Address	0x00000058
Absolute Address	sd0: 0xE0100058 sd1: 0xE0101058
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	ADMA system address register

Register ADMA_system_address Details

Field Name	Bits	Type	Reset Value	Description
ADMA_System_Address	31:0	rw	0x0	<p>This register holds byte address of executing command of the Descriptor table.</p> <p>32-bit Address Descriptor uses lower 32-bit of this register. At the start of ADMA, the Host Driver shall set start address of the Descriptor table. The ADMA increments this register address, which points to next line, when every fetching a Descriptor line. When the ADMA Error Interrupt is generated, this register shall hold valid Descriptor address depending on the ADMA state. The Host Driver shall program Descriptor Table on 32-bit boundary and set 32-bit boundary address to this register. ADMA2 ignores lower 2-bit of this register and assumes it to be 00b.</p> <p>32-bit Address ADMA Register Value 32-bit System Address</p> <p>0x00000000 0x00000000</p> <p>0x00000004 0x00000004</p> <p>to</p> <p>0xFFFFFFFFC 0xFFFFFFFFC</p>

Register ([sdio](#)) Boot_Timeout_control

Name	Boot_Timeout_control
Relative Address	0x00000060
Absolute Address	sd0: 0xE0100060 sd1: 0xE0101060
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Boot Timeout control register

Register Boot_Timeout_control Details

Field Name	Bits	Type	Reset Value	Description
Boot_Data_Timeout_Counter_Value	31:0	rw	0x0	This value determines the interval by which DAT line time-outs are detected during boot operation for MMC3.31 card. The value is in number of sd clock.

Register ([sdio](#)) Debug_Selection

Name	Debug_Selection
Relative Address	0x00000064
Absolute Address	sd0: 0xE0100064 sd1: 0xE0101064
Width	1 bits
Access Type	wo
Reset Value	0x00000000
Description	Debug Selection Register

Register Debug_Selection Details

Field Name	Bits	Type	Reset Value	Description
Debug_sel	0	wo	0x0	1- cmd register, Interrupt status, transmitter module, ahb_iface module and clk sdcard signals are probed out. 0 - receiver module and fifo_ctrl module signals are probed out

Register ([sdio](#)) SPI_interrupt_support

Name	SPI_interrupt_support
Relative Address	0x000000F0
Absolute Address	sd0: 0xE01000F0 sd1: 0xE01010F0
Width	8 bits
Access Type	rw
Reset Value	0x00000000
Description	SPI interrupt support register

Register SPI_interrupt_support Details

Field Name	Bits	Type	Reset Value	Description
SPI_INT_SUPPORT	7:0	rw	0x0	This bit is set to indicate the assertion of interrupts in the SPI mode at any time, irrespective of the status of the card select (CS) line. If this bit is zero, then SDIO card can only assert the interrupt line in the SPI mode when the CS line is asserted.

Register ([sdio](#)) Slot_interrupt_status_Host_controller_version

Name	Slot_interrupt_status_Host_controller_version
Relative Address	0x000000FC
Absolute Address	sd0: 0xE01000FC sd1: 0xE01010FC
Width	32 bits
Access Type	ro
Reset Value	0x89010000
Description	Slot interrupt status register and Host controller version register

Register Slot_interrupt_status_Host_controller_version Details

Field Name	Bits	Type	Reset Value	Description
Specification_Version_Number	31:24	ro	0x89	This status indicates the Host Controller Spec. Version. The upper and lower 4-bits indicate the version. 00 - SD Host Specification version 1.0 01 - SD Host Specification version 2.00 including only the feature of the Test Register others - Reserved
Vendor_Version_Number	23:16	ro	0x1	This status is reserved for the vendor version number. The HD should not use this status.

Field Name	Bits	Type	Reset Value	Description
reserved	15:8	ro	0x0	Reserved
Interrupt_Signal_for_Each_Slot	7:0	ro	0x0	<p>These status bit indicate the logical OR of Interrupt signal and Wakeup signal for each slot. A maximum of 8 slots can be defined. If one interrupt signal is associated with multiple slots. the HD can know which interrupt is generated by reading these status bits. By a power on reset or by Software Reset For All, the Interrupt signal shall be de asserted and this status shall read 00h.</p> <p>Bit 00 - Slot 1 Bit 01 - Slot 2 Bit 02 - Slot 3 ----- Bit 07 - Slot 8</p>

B.28 System Level Control Registers (slcr)

Module Name	System Level Control Registers (slcr)
Base Address	0xF8000000 slcr
Description	System Level Control Registers
Version	1.0
Doc Version	1.3, based on 11/18/2010 SLCR_spec.doc
Vendor Info	Xilinx Zynq slcr

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
SCL	0x00000000	32	rw	0x00000000	Secure Configuration Lock
SLCR_LOCK	0x00000004	32	wo	0x00000000	SLCR Write Protection Lock
SLCR_UNLOCK	0x00000008	32	wo	0x00000000	SLCR Write Protection Unlock
SLCR_LOCKSTA	0x0000000C	32	ro	0x00000001	SLCR Write Protection Status
ARM_PLL_CTRL	0x00000100	32	rw	0x0001A008	ARM PLL Control
DDR_PLL_CTRL	0x00000104	32	rw	0x0001A008	DDR PLL Control
IO_PLL_CTRL	0x00000108	32	rw	0x0001A008	IO PLL Control
PLL_STATUS	0x0000010C	32	ro	0x0000003F	PLL Status
ARM_PLL_CFG	0x00000110	32	rw	0x00177EA0	ARM PLL Configuration
DDR_PLL_CFG	0x00000114	32	rw	0x00177EA0	DDR PLL Configuration
IO_PLL_CFG	0x00000118	32	rw	0x00177EA0	IO PLL Configuration
ARM_CLK_CTRL	0x00000120	32	rw	0x1F000400	CPU Clock Control
DDR_CLK_CTRL	0x00000124	32	rw	0x18400003	DDR Clock Control
DCI_CLK_CTRL	0x00000128	32	rw	0x01E03201	DCI clock control
APER_CLK_CTRL	0x0000012C	32	rw	0x01FFCCCD	AMBA Peripheral Clock Control
USB0_CLK_CTRL	0x00000130	32	rw	0x00101941	USB 0 ULPI Clock Control
USB1_CLK_CTRL	0x00000134	32	rw	0x00101941	USB 1 ULPI Clock Control
GEM0_RCLK_CTRL	0x00000138	32	rw	0x00000001	GigE 0 Rx Clock Control
GEM1_RCLK_CTRL	0x0000013C	32	rw	0x00000001	GigE 1 Rx Clock Control
GEM0_CLK_CTRL	0x00000140	32	rw	0x00003C01	GigE 0 Ref Clock Control
GEM1_CLK_CTRL	0x00000144	32	rw	0x00003C01	GigE 1 Ref Clock Control
SMC_CLK_CTRL	0x00000148	32	rw	0x00003C21	SMC Ref Clock Control

Register Name	Address	Width	Type	Reset Value	Description
LQSPI_CLK_CTRL	0x0000014C	32	rw	0x00002821	Quad SPI Ref Clock Control
SDIO_CLK_CTRL	0x00000150	32	rw	0x00001E03	SDIO Ref Clock Control
UART_CLK_CTRL	0x00000154	32	rw	0x00003F03	UART Ref Clock Control
SPI_CLK_CTRL	0x00000158	32	rw	0x00003F03	SPI Ref Clock Control
CAN_CLK_CTRL	0x0000015C	32	rw	0x00501903	CAN Ref Clock Control
CAN_MIOCLK_CTRL	0x00000160	32	rw	0x00000000	CAN MIO Clock Control
DBG_CLK_CTRL	0x00000164	32	rw	0x00000F03	SoC Debug Clock Control
PCAP_CLK_CTRL	0x00000168	32	rw	0x00000F01	PCAP Clock Control
TOPSW_CLK_CTRL	0x0000016C	32	rw	0x00000000	Top-Level Switch Clock Control
FPGA0_CLK_CTRL	0x00000170	32	rw	0x00101800	PL Clock 0 Output control
FPGA1_CLK_CTRL	0x00000180	32	rw	0x00101800	PL Clock 1 Output control
FPGA2_CLK_CTRL	0x00000190	32	rw	0x00101800	PL Clock 2 output control
FPGA3_CLK_CTRL	0x000001A0	32	rw	0x00101800	PL Clock 3 output control
CLK_621_TRUE	0x000001C4	32	rw	0x00000001	CPU Clock Ratio Mode select
PSS_RST_CTRL	0x00000200	32	rw	0x00000000	PS Software Reset Control
DDR_RST_CTRL	0x00000204	32	rw	0x00000000	DDR Software Reset Control
TOPSW_RST_CTRL	0x00000208	32	rw	0x00000000	TOPSW Software Reset Control
DMAC_RST_CTRL	0x0000020C	32	rw	0x00000000	DMA Controller SW Reset Control
USB_RST_CTRL	0x00000210	32	rw	0x00000000	USB Software Reset Control
GEM_RST_CTRL	0x00000214	32	rw	0x00000000	Gigabit Ethernet SW Reset Control
SDIO_RST_CTRL	0x00000218	32	rw	0x00000000	SDIO Software Reset Control
SPI_RST_CTRL	0x0000021C	32	rw	0x00000000	SPI Software Reset Control
CAN_RST_CTRL	0x00000220	32	rw	0x00000000	CAN Software Reset Control
I2C_RST_CTRL	0x00000224	32	rw	0x00000000	I2C Software Reset Control
UART_RST_CTRL	0x00000228	32	rw	0x00000000	UART Software Reset Control
GPIO_RST_CTRL	0x0000022C	32	rw	0x00000000	GPIO Software Reset Control
LQSPI_RST_CTRL	0x00000230	32	rw	0x00000000	Quad SPI Software Reset Control
SMC_RST_CTRL	0x00000234	32	rw	0x00000000	SMC Software Reset Control
OCM_RST_CTRL	0x00000238	32	rw	0x00000000	OCM Software Reset Control
DEVCI_RST_CTRL	0x0000023C	32	rw	0x00000000	Device Config Interface SW Reset Control
FPGA_RST_CTRL	0x00000240	32	rw	0x01F33F0F	FPGA Software Reset Control

Register Name	Address	Width	Type	Reset Value	Description
A9_CPU_RST_CTRL	0x00000244	32	rw	0x00000000	CPU Software Reset Control
RS_AWDT_CTRL	0x0000024C	32	rw	0x00000000	Watchdog Timer Reset Control
REBOOT_STATUS	0x00000258	32	rw	0x00400000	Reboot Status, persistent
BOOT_MODE	0x0000025C	32	mixed	x	Boot Mode Strapping Pins
APU_CTRL	0x00000300	32	rw	0x00000000	APU Control
WDT_CLK_SEL	0x00000304	32	rw	0x00000000	APU watchdog timer clock select
PSS_IDCODE	0x00000530	32	ro	x	PS IDCODE
DDR_URGENT	0x00000600	32	rw	0x00000000	DDR Urgent Control
DDR_CAL_START	0x0000060C	32	mixed	0x00000000	DDR Calibration Start Triggers
DDR_REF_START	0x00000614	32	mixed	0x00000000	DDR Refresh Start Triggers
DDR_CMD_STA	0x00000618	32	mixed	0x00000000	DDR Command Store Status
DDR_URGENT_SEL	0x0000061C	32	rw	0x00000000	DDR Urgent Select
DDR_DFI_STATUS	0x00000620	32	mixed	0x00000000	DDR DFI status
MIO_PIN_00	0x00000700	32	rw	0x00001601	MIO Pin 0 Control
MIO_PIN_01	0x00000704	32	rw	0x00001601	MIO Pin 1 Control
MIO_PIN_02	0x00000708	32	rw	0x00000601	MIO Pin 2 Control
MIO_PIN_03	0x0000070C	32	rw	0x00000601	MIO Pin 3 Control
MIO_PIN_04	0x00000710	32	rw	0x00000601	MIO Pin 4 Control
MIO_PIN_05	0x00000714	32	rw	0x00000601	MIO Pin 5 Control
MIO_PIN_06	0x00000718	32	rw	0x00000601	MIO Pin 6 Control
MIO_PIN_07	0x0000071C	32	rw	0x00000601	MIO Pin 7 Control
MIO_PIN_08	0x00000720	32	rw	0x00000601	MIO Pin 8 Control
MIO_PIN_09	0x00000724	32	rw	0x00001601	MIO Pin 9 Control
MIO_PIN_10	0x00000728	32	rw	0x00001601	MIO Pin 10 Control
MIO_PIN_11	0x0000072C	32	rw	0x00001601	MIO Pin 11 Control
MIO_PIN_12	0x00000730	32	rw	0x00001601	MIO Pin 12 Control
MIO_PIN_13	0x00000734	32	rw	0x00001601	MIO Pin 13 Control
MIO_PIN_14	0x00000738	32	rw	0x00001601	MIO Pin 14 Control
MIO_PIN_15	0x0000073C	32	rw	0x00001601	MIO Pin 15 Control
MIO_PIN_16	0x00000740	32	rw	0x00001601	MIO Pin 16 Control
MIO_PIN_17	0x00000744	32	rw	0x00001601	MIO Pin 17 Control
MIO_PIN_18	0x00000748	32	rw	0x00001601	MIO Pin 18 Control
MIO_PIN_19	0x0000074C	32	rw	0x00001601	MIO Pin 19 Control

Register Name	Address	Width	Type	Reset Value	Description
MIO_PIN_20	0x00000750	32	rw	0x00001601	MIO Pin 20 Control
MIO_PIN_21	0x00000754	32	rw	0x00001601	MIO Pin 21 Control
MIO_PIN_22	0x00000758	32	rw	0x00001601	MIO Pin 22 Control
MIO_PIN_23	0x0000075C	32	rw	0x00001601	MIO Pin 23 Control
MIO_PIN_24	0x00000760	32	rw	0x00001601	MIO Pin 24 Control
MIO_PIN_25	0x00000764	32	rw	0x00001601	MIO Pin 25 Control
MIO_PIN_26	0x00000768	32	rw	0x00001601	MIO Pin 26 Control
MIO_PIN_27	0x0000076C	32	rw	0x00001601	MIO Pin 27 Control
MIO_PIN_28	0x00000770	32	rw	0x00001601	MIO Pin 28 Control
MIO_PIN_29	0x00000774	32	rw	0x00001601	MIO Pin 29 Control
MIO_PIN_30	0x00000778	32	rw	0x00001601	MIO Pin 30 Control
MIO_PIN_31	0x0000077C	32	rw	0x00001601	MIO Pin 31 Control
MIO_PIN_32	0x00000780	32	rw	0x00001601	MIO Pin 32 Control
MIO_PIN_33	0x00000784	32	rw	0x00001601	MIO Pin 33 Control
MIO_PIN_34	0x00000788	32	rw	0x00001601	MIO Pin 34 Control
MIO_PIN_35	0x0000078C	32	rw	0x00001601	MIO Pin 35 Control
MIO_PIN_36	0x00000790	32	rw	0x00001601	MIO Pin 36 Control
MIO_PIN_37	0x00000794	32	rw	0x00001601	MIO Pin 37 Control
MIO_PIN_38	0x00000798	32	rw	0x00001601	MIO Pin 38 Control
MIO_PIN_39	0x0000079C	32	rw	0x00001601	MIO Pin 39 Control
MIO_PIN_40	0x000007A0	32	rw	0x00001601	MIO Pin 40 Control
MIO_PIN_41	0x000007A4	32	rw	0x00001601	MIO Pin 41 Control
MIO_PIN_42	0x000007A8	32	rw	0x00001601	MIO Pin 42 Control
MIO_PIN_43	0x000007AC	32	rw	0x00001601	MIO Pin 43 Control
MIO_PIN_44	0x000007B0	32	rw	0x00001601	MIO Pin 44 Control
MIO_PIN_45	0x000007B4	32	rw	0x00001601	MIO Pin 45 Control
MIO_PIN_46	0x000007B8	32	rw	0x00001601	MIO Pin 46 Control
MIO_PIN_47	0x000007BC	32	rw	0x00001601	MIO Pin 47 Control
MIO_PIN_48	0x000007C0	32	rw	0x00001601	MIO Pin 48 Control
MIO_PIN_49	0x000007C4	32	rw	0x00001601	MIO Pin 49 Control
MIO_PIN_50	0x000007C8	32	rw	0x00001601	MIO Pin 50 Control
MIO_PIN_51	0x000007CC	32	rw	0x00001601	MIO Pin 51 Control
MIO_PIN_52	0x000007D0	32	rw	0x00001601	MIO Pin 52 Control
MIO_PIN_53	0x000007D4	32	rw	0x00001601	MIO Pin 53 Control

Register Name	Address	Width	Type	Reset Value	Description
MIO_FMIO_GEM_SEL	0x00000800	32	rw	0x00000000	Ethernet Routing Select
MIO_LOOPBACK	0x00000804	32	rw	0x00000000	Loopback function within MIO
MIO_MST_TRI0	0x0000080C	32	rw	0xFFFFFFFF	MIO pin Tri-state Enables, 31:0
MIO_MST_TRI1	0x00000810	32	rw	0x003FFFFFFF	MIO pin Tri-state Enables, 53:32
SD0_WP_CD_SEL	0x00000830	32	rw	0x00000000	SDIO 0 WP CD select
SD1_WP_CD_SEL	0x00000834	32	rw	0x00000000	SDIO 1 WP CD select
LVL_SHFTR_EN	0x00000900	32	rw	0x00000000	Level Shifters Enable
OCM_CFG	0x00000910	32	rw	0x00000000	OCM Address Mapping
GPIOB_CTRL	0x00000B00	32	rw	0x00000000	PS IO Buffer Control
GPIOB_CFG_CMOS18	0x00000B04	32	rw	0x00000000	MIO GPIOB CMOS 1.8V config
GPIOB_CFG_CMOS25	0x00000B08	32	rw	0x00000000	MIO GPIOB CMOS 2.5V config
GPIOB_CFG_CMOS33	0x00000B0C	32	rw	0x00000000	MIO GPIOB CMOS 3.3V config
GPIOB_CFG_LVTTL	0x00000B10	32	rw	0x00000000	MIO GPIOB LVTTL config
GPIOB_CFG_HSTL	0x00000B14	32	rw	0x00000000	MIO GPIOB HSTL config
GPIOB_DRV_R_BIAS_CTRL	0x00000B18	32	mixed	0x00000000	MIO GPIOB Driver Bias Control
DDRIOB_ADDR0	0x00000B40	32	rw	0x00000800	DDR IOB Config for Address 0
DDRIOB_ADDR1	0x00000B44	32	rw	0x00000800	DDR IOB Config for Address 1
DDRIOB_DATA0	0x00000B48	32	rw	0x00000800	DDR IOB Config for Data 15:0
DDRIOB_DATA1	0x00000B4C	32	rw	0x00000800	DDR IOB Config for Data 31:16
DDRIOB_DIFF0	0x00000B50	32	rw	0x00000800	DDR IOB Config for DQS 1:0
DDRIOB_DIFF1	0x00000B54	32	rw	0x00000800	DDR IOB Config for DQS 3:2
DDRIOB_CLOCK	0x00000B58	32	rw	0x00000800	DDR IOB Config for Clock Output
DDRIOB_DRIVE_SLEW_ADDR	0x00000B5C	32	rw	0x00000000	DDR IOB Slew for Address
DDRIOB_DRIVE_SLEW_DATA	0x00000B60	32	rw	0x00000000	DDR IOB Slew for Data
DDRIOB_DRIVE_SLEW_DIFF	0x00000B64	32	rw	0x00000000	DDR IOB Slew for Diff
DDRIOB_DRIVE_SLEW_CLOCK	0x00000B68	32	rw	0x00000000	DDR IOB Slew for Clock
DDRIOB_DDR_CTRL	0x00000B6C	32	rw	0x00000000	DDR IOB Buffer Control
DDRIOB_DCI_CTRL	0x00000B70	32	rw	0x00000020	DDR IOB DCI Config
DDRIOB_DCI_STATUS	0x00000B74	32	mixed	0x00000000	DDR IO Buffer DCI Status

Register ([slcr](#)) SCL

Name	SCL
Relative Address	0x00000000
Absolute Address	0xF8000000
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Secure Configuration Lock

Register SCL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
LOCK	0	rw	0x0	Secure configuration lock: 0: unlocked, Secure writes to Secure configuration registers are enabled. 1: locked, all writes to Secure configuration registers are disabled and cannot be enabled until power-on reset is asserted. (This includes registers at the following address 0x0, 0x200, 0x300, 0x304, 0x400-0x488, 0x500 within SLCR address space.) This field is reset by POR only. .

Register ([slcr](#)) SLCR_LOCK

Name	SLCR_LOCK
Relative Address	0x00000004
Absolute Address	0xF8000004
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	SLCR Write Protection Lock

Register SLCR_LOCK Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	wo	0x0	Reserved. Writes are ignored, read data is zero.
LOCK_KEY	15:0	wo	0x0	When write data contains the lock key value of 0x767B, the write protection mode is enabled. All registers defined in SLCR are write protected until unlocked again through the SLCR_UNLOCK register. A read of this register returns zero. (This covers all registers in the SLCR register space)

Register ([slcr](#)) SLCR_UNLOCK

Name	SLCR_UNLOCK
Relative Address	0x00000008
Absolute Address	0xF8000008
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	SLCR Write Protection Unlock

Register SLCR_UNLOCK Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	wo	0x0	Reserved. Writes are ignored, read data is zero.
UNLOCK_KEY	15:0	wo	0x0	When write data contains the unlock key value of 0xDF0D, the write protection mode is disabled. All registers defined in SLCR are writeable until locked again through the SLCR_LOCK register. A read of this register returns zero.

Register ([slcr](#)) SLCR_LOCKSTA

Name	SLCR_LOCKSTA
Relative Address	0x0000000C
Absolute Address	0xF800000C
Width	32 bits
Access Type	ro
Reset Value	0x00000001
Description	SLCR Write Protection Status

Register SLCR_LOCKSTA Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	ro	0x0	Reserved. Writes are ignored, read data is zero.
LOCK_STATUS	0	ro	0x1	Current state of write protection mode of SLCR: 0: Registers are writeable 1: Registers are not writeable. Any attempted writes are ignored, but reads will complete as normal.

Register ([slcr](#)) ARM_PLL_CTRL

Name	ARM_PLL_CTRL
Relative Address	0x00000100
Absolute Address	0xF8000100
Width	32 bits
Access Type	rw
Reset Value	0x0001A008
Description	ARM PLL Control

Register ARM_PLL_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:19	rw	0x0	Reserved. Writes are ignored, read data is zero.
PLL_FDIV	18:12	rw	0x1A	Provide the feedback divisor for the PLL. Note: Before changing this value, the PLL must first be bypassed and then put into reset. (Remember to set the appropriate CP/RES/CNT values. Please see TRM)
reserved	11:5	rw	0x0	Reserved. Writes are ignored, read data is zero.
PLL_BYPASS_FORCE	4	rw	0x0	ARM PLL Bypass override control: PLL_BYPASS_QUAL = 0: 0: enabled, not bypassed. 1: bypassed. PLL_BYPASS_QUAL = 1: 0: 1: bypass mode regardless of the pin strapping.

Field Name	Bits	Type	Reset Value	Description
PLL_BYPASS_QUAL	3	rw	0x1	Select the source for the ARM PLL Bypass Control: 0: controlled by the PLL_BYPASS_FORCE bit, bit 4. 1: controlled by the value of the sampled BOOT_MODE pin strapping resistor PLL_BYPASS. This can be read using the BOOT_MODE[4] bit.
reserved	2	rw	0x0	Reserved. Writes are ignored, read data is zero.
PLL_PWRDWN	1	rw	0x0	Drives the PWRDWN input of the PLL: 0: PLL powered up 1: PLL powered down
PLL_RESET	0	rw	0x0	PLL reset control: 0: PLL out of reset 1: PLL held in reset.

Register ([slcr](#)) DDR_PLL_CTRL

Name	DDR_PLL_CTRL
Relative Address	0x00000104
Absolute Address	0xF8000104
Width	32 bits
Access Type	rw
Reset Value	0x0001A008
Description	DDR PLL Control

Register DDR_PLL_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:19	rw	0x0	Reserved. Writes are ignored, read data is zero.
PLL_FDIV	18:12	rw	0x1A	Provide the feedback divisor for the PLL. Note: Before changing this value, the PLL must first be bypassed and then put into reset. (Remember to set the appropriate CP/RES/CNT values. Please see TRM)
reserved	11:5	rw	0x0	Reserved. Writes are ignored, read data is zero.
PLL_BYPASS_FORCE	4	rw	0x0	Override control of the PLL bypass function within the clock controller to force into bypass state: 0: PLL not forced to be bypassed (may still be bypassed through bootstrap pin). 1: PLL forced to be bypassed

Field Name	Bits	Type	Reset Value	Description
PLL_BYPASS_QUAL	3	rw	0x1	Qualify the PLL bootstrap signal sampled from boot_mode[3] in the MIO. By default, the bootstrap will directly control all three PLL bypass muxes within the clock controller. 0: PLL bypass mux in clock controller is only controlled by PLL_BYPASS_FORCE bit. 1: PLL bypass mux in clock controller is controlled by sampled boot_mode[3] provided PLL_BYPASS_FORCE is not set.
reserved	2	rw	0x0	Reserved. Writes are ignored, read data is zero.
PLL_PWRDWN	1	rw	0x0	Drive the PWRDWN input of the PLL: 0: PLL powered up 1: PLL powered down
PLL_RESET	0	rw	0x0	PLL reset control: 0: PLL out of reset 1: PLL held in reset.

Register ([slcr](#)) IO_PLL_CTRL

Name	IO_PLL_CTRL
Relative Address	0x00000108
Absolute Address	0xF8000108
Width	32 bits
Access Type	rw
Reset Value	0x0001A008
Description	IO PLL Control

Register IO_PLL_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:19	rw	0x0	Reserved. Writes are ignored, read data is zero.
PLL_FDIV	18:12	rw	0x1A	Provide the feedback divisor for the PLL. Note: Before changing this value, the PLL must first be bypassed and then put into reset. (Remember to set the appropriate CP/RES/CNT values. Please see TRM)
reserved	11:5	rw	0x0	Reserved. Writes are ignored, read data is zero.

Field Name	Bits	Type	Reset Value	Description
PLL_BYPASS_FORCE	4	rw	0x0	Override control of the PLL bypass function within the clock controller to force into bypass state: 0: PLL not forced to be bypassed (may still be bypassed through bootstrap pin). 1: PLL forced to be bypassed
PLL_BYPASS_QUAL	3	rw	0x1	Qualify the PLL bootstrap signal sampled from boot_mode[3] in the MIO. By default, the bootstrap will directly control all three PLL bypass muxes within the clock controller. 0: PLL bypass mux in clock controller is only controlled by PLL_BYPASS_FORCE bit. 1: PLL bypass mux in clock controller is controlled by sampled boot_mode[3] provided PLL_BYPASS_FORCE is not set.
reserved	2	rw	0x0	Reserved. Writes are ignored, read data is zero.
PLL_PWRDWN	1	rw	0x0	Drive the PWRDWN input of the PLL: 0: PLL powered up 1: PLL powered down
PLL_RESET	0	rw	0x0	PLL reset control: 0: PLL out of reset 1: PLL held in reset.

Register ([slcr](#)) PLL_STATUS

Name	PLL_STATUS
Relative Address	0x0000010C
Absolute Address	0xF800010C
Width	32 bits
Access Type	ro
Reset Value	0x0000003F
Description	PLL Status

Register PLL_STATUS Details

Note: Reset condition is actually 0, but will read a 1 by the time this register can be read by software if PLLs are enabled by BOOT_MODE.

Field Name	Bits	Type	Reset Value	Description
reserved	31:6	ro	0x0	Reserved. Writes are ignored, read data is zero.
IO_PLL_STABLE	5	ro	0x1	IO PLL clock stable status: 0: not locked and not in bypass 1: locked or bypassed

Field Name	Bits	Type	Reset Value	Description
DDR_PLL_STABLE	4	ro	0x1	DDR PLL clock stable status: 0: not locked and not in bypass 1: locked or bypassed
ARM_PLL_STABLE	3	ro	0x1	ARM PLL clock stable status: 0: not locked and not in bypass 1: locked or bypassed
IO_PLL_LOCK	2	ro	0x1	IO PLL lock status: 0: not locked, 1: locked
DDR_PLL_LOCK	1	ro	0x1	DDR PLL lock status: 0: not locked, 1: locked
ARM_PLL_LOCK	0	ro	0x1	ARM PLL lock status: 0: not locked, 1: locked

Register ([slcr](#)) ARM_PLL_CFG

Name	ARM_PLL_CFG
Relative Address	0x00000110
Absolute Address	0xF8000110
Width	32 bits
Access Type	rw
Reset Value	0x00177EA0
Description	ARM PLL Configuration

Register ARM_PLL_CFG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rw	0x0	Reserved. Writes are ignored, read data is zero.
LOCK_CNT	21:12	rw	0x177	Drive the LOCK_CNT[9:0] input of the PLL to set the number of clock cycles the PLL needs to have clkref and clkfb aligned with a certain window before syaing locked.
PLL_CP	11:8	rw	0xE	Drive the PLL_CP[3:0] input of the PLL to set the PLL charge pump control
PLL_RES	7:4	rw	0xA	Drive the PLL_RES[3:0] input of the PLL to set the PLL loop filter resistor control
reserved	3:0	rw	0x0	Reserved. Writes are ignored, read data is zero.

Register ([slcr](#)) DDR_PLL_CFG

Name	DDR_PLL_CFG
Relative Address	0x00000114
Absolute Address	0xF8000114
Width	32 bits
Access Type	rw
Reset Value	0x00177EA0
Description	DDR PLL Configuration

Register DDR_PLL_CFG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rw	0x0	Reserved. Writes are ignored, read data is zero.
LOCK_CNT	21:12	rw	0x177	Drive the LOCK_CNT[9:0] input of the PLL to set the number of clock cycles the PLL needs to have clkref and clkfb aligned with a certain window before staying locked.
PLL_CP	11:8	rw	0xE	Drive the PLL_CP[3:0] input of the PLL to set the PLL charge pump control.
PLL_RES	7:4	rw	0xA	Drive the PLL_RES[3:0] input of the PLL to set the PLL loop filter resistor control.
reserved	3:0	rw	0x0	Reserved. Writes are ignored, read data is zero.

Register ([slcr](#)) IO_PLL_CFG

Name	IO_PLL_CFG
Relative Address	0x00000118
Absolute Address	0xF8000118
Width	32 bits
Access Type	rw
Reset Value	0x00177EA0
Description	IO PLL Configuration

Register IO_PLL_CFG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rw	0x0	Reserved. Writes are ignored, read data is zero.
LOCK_CNT	21:12	rw	0x177	Drive the LOCK_CNT[9:0] input of the PLL to set the number of clock cycles the PLL needs to have clkref and clkfb aligned with a certain window before staying locked.
PLL_CP	11:8	rw	0xE	Drive the PLL_CP[3:0] input of the PLL to set the PLL charge pump control.
PLL_RES	7:4	rw	0xA	Drive the PLL_RES[3:0] input of the PLL to set the PLL loop filter resistor control.
reserved	3:0	rw	0x0	Reserved. Writes are ignored, read data is zero.

Register ([slcr](#)) ARM_CLK_CTRL

Name	ARM_CLK_CTRL
Relative Address	0x00000120
Absolute Address	0xF8000120
Width	32 bits
Access Type	rw
Reset Value	0x1F000400
Description	CPU Clock Control

Register ARM_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:29	rw	0x0	Reserved. Writes are ignored, read data is zero.
CPU_PERI_CLKACT	28	rw	0x1	Clock active: 0: Clock is disabled 1: Clock is enabled
CPU_1XCLKACT	27	rw	0x1	CPU_1x Clock control: 0: disable, 1: enable
CPU_2XCLKACT	26	rw	0x1	CPU_2x Clock control: 0: disable, 1: enable
CPU_3OR2XCLKACT	25	rw	0x1	CPU_3x2x Clock control: 0: disable, 1: enable
CPU_6OR4XCLKACT	24	rw	0x1	CPU_6x4x Clock control: 0: disable, 1: enable
reserved	23:14	rw	0x0	Reserved. Writes are ignored, read data is zero.

Field Name	Bits	Type	Reset Value	Description
DIVISOR	13:8	rw	0x4	Frequency divisor for the CPU clock source.
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	5:4	rw	0x0	Select the source used to generate the CPU clock: 0x: ARM PLL 10: DDR PLL 11: IO PLL This field is reset by POR only.
reserved	3:0	rw	0x0	Reserved. Writes are ignored, read data is zero.

Register ([slcr](#)) DDR_CLK_CTRL

Name	DDR_CLK_CTRL
Relative Address	0x00000124
Absolute Address	0xF8000124
Width	32 bits
Access Type	rw
Reset Value	0x18400003
Description	DDR Clock Control

Register DDR_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
DDR_2XCLK_DIVISOR	31:26	rw	0x6	Frequency divisor for the ddr_2x clock
DDR_3XCLK_DIVISOR	25:20	rw	0x4	Frequency divisor for the ddr_3x clock
reserved	19:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
DDR_2XCLKACT	1	rw	0x1	DDR_2x Clock control: 0: disable, 1: enable
DDR_3XCLKACT	0	rw	0x1	DDR_3x Clock control: 0: disable, 1: enable

Register ([slcr](#)) DCI_CLK_CTRL

Name	DCI_CLK_CTRL
Relative Address	0x00000128
Absolute Address	0xF8000128

Width	32 bits
Access Type	rw
Reset Value	0x01E03201
Description	DCI clock control

Register DCI_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR1	25:20	rw	0x1E	Provides the divisor used to divide the source clock to generate the required generated clock frequency. Second cascade divider
reserved	19:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR0	13:8	rw	0x32	Provides the divisor used to divide the source clock to generate the required generated clock frequency.
reserved	7:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLKACT	0	rw	0x1	DCI clock control - 0: disable, 1: enable

Register ([slcr](#)) APER_CLK_CTRL

Name	APER_CLK_CTRL
Relative Address	0x0000012C
Absolute Address	0xF800012C
Width	32 bits
Access Type	rw
Reset Value	0x01FFCCCD
Description	AMBA Peripheral Clock Control

Register APER_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:25	rw	0x0	Reserved. Writes are ignored, read data is zero.
SMC_CPU_1XCLKACT	24	rw	0x1	SMC AMBA Clock control 0: disable, 1: enable
LQSPI_CPU_1XCLKACT	23	rw	0x1	Quad SPI AMBA Clock control 0: disable, 1: enable
GPIO_CPU_1XCLKACT	22	rw	0x1	GPIO AMBA Clock control 0: disable, 1: enable

Field Name	Bits	Type	Reset Value	Description
UART1_CPU_1XCLKACT	21	rw	0x1	UART 1 AMBA Clock control 0: disable, 1: enable
UART0_CPU_1XCLKACT	20	rw	0x1	UART 0 AMBA Clock control 0: disable, 1: enable
I2C1_CPU_1XCLKACT	19	rw	0x1	I2C 1 AMBA Clock control 0: disable, 1: enable
I2C0_CPU_1XCLKACT	18	rw	0x1	I2C 0 AMBA Clock control 0: disable, 1: enable
CAN1_CPU_1XCLKACT	17	rw	0x1	CAN 1 AMBA Clock control 0: disable, 1: enable
CAN0_CPU_1XCLKACT	16	rw	0x1	CAN 0 AMBA Clock control 0: disable, 1: enable
SPI1_CPU_1XCLKACT	15	rw	0x1	SPI 1 AMBA Clock control 0: disable, 1: enable
SPI0_CPU_1XCLKACT	14	rw	0x1	SPI 0 AMBA Clock control 0: disable, 1: enable
reserved	13	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	12	rw	0x0	Reserved. Writes are ignored, read data is zero.
SDI1_CPU_1XCLKACT	11	rw	0x1	SDIO controller 1 AMBA Clock control 0: disable, 1: enable
SDI0_CPU_1XCLKACT	10	rw	0x1	SDIO controller 0 AMBA Clock 0: disable, 1: enable
reserved	9	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	8	rw	0x0	Reserved. Writes are ignored, read data is zero.
GEM1_CPU_1XCLKACT	7	rw	0x1	Gigabit Ethernet 1 AMBA Clock control 0: disable, 1: enable
GEM0_CPU_1XCLKACT	6	rw	0x1	Gigabit Ethernet 0 AMBA Clock control 0: disable, 1: enable
reserved	5	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	4	rw	0x0	Reserved. Writes are ignored, read data is zero.
USB1_CPU_1XCLKACT	3	rw	0x1	USB controller 1 AMBA Clock control 0: disable, 1: enable
USB0_CPU_1XCLKACT	2	rw	0x1	USB controller 0 AMBA Clock control 0: disable, 1: enable

Field Name	Bits	Type	Reset Value	Description
reserved	1	rw	0x0	Reserved. Writes are ignored, read data is zero.
DMA_CPU_2XCLKACT	0	rw	0x1	DMA controller AMBA Clock control 0: disable, 1: enable

Register ([slcr](#)) USB0_CLK_CTRL

Name	USB0_CLK_CTRL
Relative Address	0x00000130
Absolute Address	0xF8000130
Width	32 bits
Access Type	rw
Reset Value	0x00101941
Description	USB 0 ULPI Clock Control

Register USB0_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	25:20	rw	0x1	Reserved. Do not modify.
reserved	19:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	13:8	rw	0x19	Reserved. Do not modify.
reserved	7	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	6:4	rw	0x4	Select the source to generate USB controller 0 ULPI clock: 1xx: USB 0 MIO ULPI clock (top level MIO ULPI clock is an input)
reserved	3:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	0	rw	0x1	Reserved. Do not modify.

Register ([slcr](#)) USB1_CLK_CTRL

Name	USB1_CLK_CTRL
Relative Address	0x00000134
Absolute Address	0xF8000134
Width	32 bits
Access Type	rw
Reset Value	0x00101941

Description USB 1 ULPI Clock Control

Register USB1_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	25:20	rw	0x1	Reserved. Do not modify.
reserved	19:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	13:8	rw	0x19	Reserved. Do not modify.
reserved	7	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	6:4	rw	0x4	Select the source to generate USB controller 1 ULPI clock: 1xx: USB 1 MIO ULPI clock (top level MIO ULPI clock is an input)
reserved	3:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	0	rw	0x1	Reserved. Do not modify.

Register ([slcr](#)) GEM0_RCLK_CTRL

Name GEM0_RCLK_CTRL
Relative Address 0x00000138
Absolute Address 0xF8000138
Width 32 bits
Access Type rw
Reset Value 0x00000001
Description GigE 0 Rx Clock Control

Register GEM0_RCLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:5	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	4	rw	0x0	Select the source to generate the Rx clock: 0: MIO Rx clock, 1: EMIO Rx clock
reserved	3:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLKACT	0	rw	0x1	Ethernet Controller 0 Rx Clock control 0: disable, 1: enable

Register ([slcr](#)) GEM1_RCLK_CTRL

Name	GEM1_RCLK_CTRL
Relative Address	0x0000013C
Absolute Address	0xF800013C
Width	32 bits
Access Type	rw
Reset Value	0x00000001
Description	GigE 1 Rx Clock Control

Register GEM1_RCLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:5	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	4	rw	0x0	Select the source to generate the Rx clock: 0: MIO Rx clock, 1: EMIO Rx clock
reserved	3:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLKACT	0	rw	0x1	Ethernet Controller 1 Rx Clock control 0: disable, 1: enable

Register ([slcr](#)) GEM0_CLK_CTRL

Name	GEM0_CLK_CTRL
Relative Address	0x00000140
Absolute Address	0xF8000140
Width	32 bits
Access Type	rw
Reset Value	0x00003C01
Description	GigE 0 Ref Clock Control

Register GEM0_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR1	25:20	rw	0x0	Second divisor for Ethernet controller 0 source clock.
reserved	19:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR	13:8	rw	0x3C	First divisor for Ethernet controller 0 source clock.
reserved	7	rw	0x0	Reserved. Writes are ignored, read data is zero.

Field Name	Bits	Type	Reset Value	Description
SRCSEL	6:4	rw	0x0	Selects the source to generate the reference clock 00x: IO PLL. 010: ARM PLL. 011: DDR PLL 1xx: Ethernet controller 0 EMIO clock
reserved	3:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLKACT	0	rw	0x1	Ethernet Controller 0 Reference Clock control 0: disable, 1: enable

Register ([slcr](#)) GEM1_CLK_CTRL

Name	GEM1_CLK_CTRL
Relative Address	0x00000144
Absolute Address	0xF8000144
Width	32 bits
Access Type	rw
Reset Value	0x00003C01
Description	GigE 1 Ref Clock Control

Register GEM1_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR1	25:20	rw	0x0	Second divisor for Ethernet controller 1 source clock.
reserved	19:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR	13:8	rw	0x3C	First divisor for Ethernet controller 1 source clock.
reserved	7	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	6:4	rw	0x0	Selects the source to generate the reference clock 00x: IO PLL. 010: ARM PLL. 011: DDR PLL 1xx: Ethernet controller 1 EMIO clock
reserved	3:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLKACT	0	rw	0x1	Ethernet Controller 1 Reference Clock control 0: disable, 1: enable

Register ([slcr](#)) SMC_CLK_CTRL

Name	SMC_CLK_CTRL
Relative Address	0x00000148
Absolute Address	0xF8000148
Width	32 bits
Access Type	rw
Reset Value	0x00003C21
Description	SMC Ref Clock Control

Register SMC_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR	13:8	rw	0x3C	Divisor for SMC source clock.
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	5:4	rw	0x2	Select clock source generate SMC clock: 0x: IO PLL, 10: ARM PLL, 11: DDR PLL
reserved	3:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLKACT	0	rw	0x1	SMC Reference Clock control 0: disable, 1: enable

Register ([slcr](#)) LQSPI_CLK_CTRL

Name	LQSPI_CLK_CTRL
Relative Address	0x0000014C
Absolute Address	0xF800014C
Width	32 bits
Access Type	rw
Reset Value	0x00002821
Description	Quad SPI Ref Clock Control

Register LQSPI_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR	13:8	rw	0x28	Divisor for Quad SPI Controller source clock.
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.

Field Name	Bits	Type	Reset Value	Description
SRCSEL	5:4	rw	0x2	Select clock source generate Quad SPI clock: 0x: IO PLL, 10: ARM PLL, 11: DDR PLL
reserved	3:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLKACT	0	rw	0x1	Quad SPI Controller Reference Clock control 0: disable, 1: enable

Register ([slcr](#)) SDIO_CLK_CTRL

Name	SDIO_CLK_CTRL
Relative Address	0x00000150
Absolute Address	0xF8000150
Width	32 bits
Access Type	rw
Reset Value	0x00001E03
Description	SDIO Ref Clock Control

Register SDIO_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR	13:8	rw	0x1E	Provides the divisor used to divide the source clock to generate the required generated clock frequency.
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	5:4	rw	0x0	Select the source used to generate the clock. 0x: Source for generated clock is IO PLL. 10: Source for generated clock is ARM PLL. 11: Source for generated clock is DDR PLL.
reserved	3:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLKACT1	1	rw	0x1	SDIO Controller 1 Clock control. 0: disable, 1: enable
CLKACT0	0	rw	0x1	SDIO Controller 0 Clock control. 0: disable, 1: enable

Register ([slcr](#)) UART_CLK_CTRL

Name	UART_CLK_CTRL
Relative Address	0x00000154

Absolute Address 0xF8000154
 Width 32 bits
 Access Type rw
 Reset Value 0x00003F03
 Description UART Ref Clock Control

Register UART_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR	13:8	rw	0x3F	Divisor for UART Controller source clock.
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	5:4	rw	0x0	Selects the PLL source to generate the clock. 0x: IO PLL 10: ARM PLL 11: DDR PLL
reserved	3:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLKACT1	1	rw	0x1	UART 1 reference clock active: 0: Clock is disabled 1: Clock is enabled
CLKACT0	0	rw	0x1	UART 0 Reference clock control. 0: disable, 1: enable

Register ([slcr](#)) SPI_CLK_CTRL

Name SPI_CLK_CTRL
 Relative Address 0x00000158
 Absolute Address 0xF8000158
 Width 32 bits
 Access Type rw
 Reset Value 0x00003F03
 Description SPI Ref Clock Control

Register SPI_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR	13:8	rw	0x3F	Provides the divisor used to divide the source clock to generate the required generated clock frequency.
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	5:4	rw	0x0	Select the source used to generate the clock: 0x: Source for generated clock is IO PLL. 10: Source for generated clock is ARM PLL. 11: Source for generated clock is DDR PLL.
reserved	3:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLKACT1	1	rw	0x1	SPI 1 reference clock active: 0: Clock is disabled 1: Clock is enabled
CLKACT0	0	rw	0x1	SPI 0 reference clock active: 0: Clock is disabled 1: Clock is enabled

Register ([slcr](#)) CAN_CLK_CTRL

Name	CAN_CLK_CTRL
Relative Address	0x0000015C
Absolute Address	0xF800015C
Width	32 bits
Access Type	rw
Reset Value	0x00501903
Description	CAN Ref Clock Control

Register CAN_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR1	25:20	rw	0x5	Provides the divisor used to divide the source clock to generate the required generated clock frequency. Second cascade divider.
reserved	19:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR0	13:8	rw	0x19	Provides the divisor used to divide the source clock to generate the required generated clock frequency. First cascade divider

Field Name	Bits	Type	Reset Value	Description
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	5:4	rw	0x0	Select the source used to generate the clock: 0x: Source for generated clock is IO PLL. 10: Source for generated clock is ARM PLL. 11: Source for generated clock is DDR PLL.
reserved	3:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLKACT1	1	rw	0x1	CAN 1 Reference Clock active: 0: Clock is disabled 1: Clock is enabled
CLKACT0	0	rw	0x1	CAN 0 Reference Clock active: 0: Clock is disabled 1: Clock is enabled

Register ([slcr](#)) CAN_MIOCLK_CTRL

Name	CAN_MIOCLK_CTRL
Relative Address	0x00000160
Absolute Address	0xF8000160
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	CAN MIO Clock Control

Register CAN_MIOCLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:23	rw	0x0	Reserved. Writes are ignored, read data is zero.
CAN1_REF_SEL	22	rw	0x0	CAN 1 Reference Clock selection: 0: From internal PLL. 1: From MIO based on the next field
CAN1_MUX	21:16	rw	0x0	CAN 1 mux selection for MIO. Setting this to zero will select MIO[0] as the clock source. Only values 0-53 are valid.
reserved	15:7	rw	0x0	Reserved. Writes are ignored, read data is zero.

Field Name	Bits	Type	Reset Value	Description
CAN0_REF_SEL	6	rw	0x0	CAN 0 Reference Clock selection: 0: From internal PLL 1: From MIO based on the next field
CAN0_MUX	5:0	rw	0x0	CAN 0 mux selection for MIO. Setting this to zero will select MIO[0] as the clock source. Only values 0-53 are valid.

Register ([slcr](#)) DBG_CLK_CTRL

Name	DBG_CLK_CTRL
Relative Address	0x00000164
Absolute Address	0xF8000164
Width	32 bits
Access Type	rw
Reset Value	0x00000F03
Description	SoC Debug Clock Control

Register DBG_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR	13:8	rw	0xF	Provides the divisor used to divide the source clock to generate the required generated debug trace clock frequency.
reserved	7	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	6:4	rw	0x0	Select the source used to generate the clock: 1xx: Source for generated clock EMIO trace clock 00x: Source for generated clock is IO PLL 010: Source for generated clock is ARM PLL 011: Source for generated clock is DDR PLL
reserved	3:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
CPU_1XCLKACT	1	rw	0x1	Debug CPU 1x Clock active. 0 - Clocks are disabled. 1 - Clocks are enabled
CLKACT_TRC	0	rw	0x1	Debug Trace Clock active: 0: Clock is disabled 1: Clock is enabled

Register ([slcr](#)) PCAP_CLK_CTRL

Name	PCAP_CLK_CTRL
Relative Address	0x00000168
Absolute Address	0xF8000168
Width	32 bits
Access Type	rw
Reset Value	0x00000F01
Description	PCAP Clock Control

Register PCAP_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR	13:8	rw	0xF	Provides the divisor used to divide the source clock to generate the required generated clock frequency.
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	5:4	rw	0x0	Select the source used to generate the clock: 0x: Source for generated clock is IO PLL. 10: Source for generated clock is ARM PLL. 11: Source for generated clock is DDR PLL.
reserved	3:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLKACT	0	rw	0x1	Clock active: 0: Clock is disabled 1: Clock is enabled

Register ([slcr](#)) TOPSW_CLK_CTRL

Name	TOPSW_CLK_CTRL
Relative Address	0x0000016C
Absolute Address	0xF800016C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Top-Level Switch Clock Control

Register TOPSW_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLK_DIS	0	rw	0x0	Clock disable control: 0: Clock is not disabled 1: Clock can be disabled

Register ([slcr](#)) FPGA0_CLK_CTRL

Name	FPGA0_CLK_CTRL
Relative Address	0x00000170
Absolute Address	0xF8000170
Width	32 bits
Access Type	rw
Reset Value	0x00101800
Description	PL Clock 0 Output control

Register FPGA0_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR1	25:20	rw	0x1	Provides the divisor used to divide the source clock to generate the required generated clock frequency. Second cascade divide
reserved	19:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR0	13:8	rw	0x18	Provides the divisor used to divide the source clock to generate the required generated clock frequency. First cascade divider.
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	5:4	rw	0x0	Select the source used to generate the clock: 0x: Source for generated clock is IO PLL. 10: Source for generated clock is ARM PLL. 11: Source for generated clock is DDR PLL.
reserved	3:0	rw	0x0	Reserved. Writes are ignored, read data is zero.

Register ([slcr](#)) FPGA1_CLK_CTRL

Name	FPGA1_CLK_CTRL
Relative Address	0x00000180

Absolute Address	0xF8000180
Width	32 bits
Access Type	rw
Reset Value	0x00101800
Description	PL Clock 1 Output control

Register FPGA1_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR1	25:20	rw	0x1	Provides the divisor used to divide the source clock to generate the required generated clock frequency. Second cascade divide
reserved	19:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR0	13:8	rw	0x18	Provides the divisor used to divide the source clock to generate the required generated clock frequency. First cascade divider.
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	5:4	rw	0x0	Select the source used to generate the clock: 0x: Source for generated clock is IO PLL. 10: Source for generated clock is ARM PLL. 11: Source for generated clock is DDR PLL.
reserved	3:0	rw	0x0	Reserved. Writes are ignored, read data is zero.

Register ([slcr](#)) FPGA2_CLK_CTRL

Name	FPGA2_CLK_CTRL
Relative Address	0x00000190
Absolute Address	0xF8000190
Width	32 bits
Access Type	rw
Reset Value	0x00101800
Description	PL Clock 2 output control

Register FPGA2_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR1	25:20	rw	0x1	Provides the divisor used to divide the source clock to generate the required generated clock frequency. Second cascade divide
reserved	19:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR0	13:8	rw	0x18	Provides the divisor used to divide the source clock to generate the required generated clock frequency. First cascade divider.
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
SRCSEL	5:4	rw	0x0	Select the source used to generate the clock: 0x: Source for generated clock is IO PLL. 10: Source for generated clock is ARM PLL. 11: Source for generated clock is DDR PLL.
reserved	3:0	rw	0x0	Reserved. Writes are ignored, read data is zero.

Register ([slcr](#)) FPGA3_CLK_CTRL

Name	FPGA3_CLK_CTRL
Relative Address	0x000001A0
Absolute Address	0xF80001A0
Width	32 bits
Access Type	rw
Reset Value	0x00101800
Description	PL Clock 3 output control

Register FPGA3_CLK_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:26	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR1	25:20	rw	0x1	Provides the divisor used to divide the source clock to generate the required generated clock frequency. Second cascade divide
reserved	19:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
DIVISOR0	13:8	rw	0x18	Provides the divisor used to divide the source clock to generate the required generated clock frequency. First cascade divider.
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.

Field Name	Bits	Type	Reset Value	Description
SRCSEL	5:4	rw	0x0	Select the source used to generate the clock: 0x: Source for generated clock is IO PLL. 10: Source for generated clock is ARM PLL. 11: Source for generated clock is DDR PLL.
reserved	3:0	rw	0x0	Reserved. Writes are ignored, read data is zero.

Register ([slcr](#)) CLK_621_TRUE

Name	CLK_621_TRUE
Relative Address	0x000001C4
Absolute Address	0xF80001C4
Width	32 bits
Access Type	rw
Reset Value	0x00000001
Description	CPU Clock Ratio Mode select

Register CLK_621_TRUE Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
CLK_621_TRUE	0	rw	0x1	Select the CPU clock ration: 0: 4:2:1 1: 6:2:1

Register ([slcr](#)) PSS_RST_CTRL

Name	PSS_RST_CTRL
Relative Address	0x00000200
Absolute Address	0xF8000200
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	PS Software Reset Control

Register PSS_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
SOFT_RST	0	rw	0x0	Processing System software reset control signal. 0: no affect 1: asserts PS software reset pulse (entire system except clock generator) There is no need to write a 0, the hardware generates a pulse everytime a 1 is written.

Register ([slcr](#)) DDR_RST_CTRL

Name	DDR_RST_CTRL
Relative Address	0x00000204
Absolute Address	0xF8000204
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	DDR Software Reset Control

Register DDR_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
DDR_RST	0	rw	0x0	DDR software reset control signal 0: disable, 1: enable

Register ([slcr](#)) TOPSW_RST_CTRL

Name	TOPSW_RST_CTRL
Relative Address	0x00000208
Absolute Address	0xF8000208
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	TOPSW Software Reset Control

Register TOPSW_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
TOPSW_RST	0	rw	0x0	TOPSW software reset control signal 0: disable, 1: enable Care must be taken to ensure that the switches do not have any outstanding transactions when using this reset.

Register ([slcr](#)) DMAC_RST_CTRL

Name	DMAC_RST_CTRL
Relative Address	0x0000020C
Absolute Address	0xF800020C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	DMA Controller SW Reset Control

Register DMAC_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
DMAC_RST	0	rw	0x0	DMA Controller software reset signal. 0: de-assert (DMA controller TrustZone register is read only) 1: assert (DMA controller TrustZone register is writeable)

Register ([slcr](#)) USB_RST_CTRL

Name	USB_RST_CTRL
Relative Address	0x00000210
Absolute Address	0xF8000210
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	USB Software Reset Control

Register USB_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
USB1_CPU1X_RST	1	rw	0x0	USB 1 master and slave AMBA software reset. On assertion of this reset, the master and slave AMBA clock portion of the USB 1 subsystem will be reset. 0: No reset 1: master and slave AMBA clock portion of USB 1 subsystem held in reset
USB0_CPU1X_RST	0	rw	0x0	USB 0 master and slave AMBA software reset. On assertion of this reset, the master and slave AMBA clock portion of the USB 0 subsystem will be reset. 0: No reset 1: master and slave AMBA clock portion of USB 0 subsystem held in reset

Register ([slcr](#)) GEM_RST_CTRL

Name	GEM_RST_CTRL
Relative Address	0x00000214
Absolute Address	0xF8000214
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Gigabit Ethernet SW Reset Control

Register GEM_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rw	0x0	Reserved. Writes are ignored, read data is zero.
GEM1_REF_RST	7	rw	0x0	Gigabit Ethernet 1 Reference software reset. On assertion of this reset, the Reference clock portion of the GEM 1 subsystem will be reset. 0: No reset 1: Reference clock portion of GEM 1 subsystem held in reset
GEM0_REF_RST	6	rw	0x0	Gigabit Ethernet 0 Reference software reset. On assertion of this reset, the Reference clock portion of the GEM 0 subsystem will be reset. 0: No reset 1: Reference clock portion of GEM 0 subsystem held in reset

Field Name	Bits	Type	Reset Value	Description
GEM1_RX_RST	5	rw	0x0	Gigabit Ethernet MAC 1 RX software reset. On assertion of this reset, the RX clock portion of the GEM 1 subsystem will be reset. 0: No reset 1: RX clock portion of GEM 1 subsystem held in reset
GEM0_RX_RST	4	rw	0x0	Gigabit Ethernet MAC 0 RX software reset. On assertion of this reset, the RX clock portion of the GEM 0 subsystem will be reset. 0: No reset 1: RX clock portion of GEM 0 subsystem held in reset
reserved	3:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
GEM1_CPU1X_RST	1	rw	0x0	Gigabit Ethernet MAC 1 software reset. On assertion of this reset, the Gigabit Ethernet MAC 1: subsystem will be reset. 0: No reset. 1: Gigabit Ethernet MAC 1 subsystem held in reset.
GEM0_CPU1X_RST	0	rw	0x0	Gigabit Ethernet MAC 0 software reset. On assertion of this reset, the Gigabit Ethernet 0 subsystem will be reset. 0: No reset 1: Gigabit Ethernet MAC 0 subsystem held in reset

Register ([slcr](#)) SDIO_RST_CTRL

Name	SDIO_RST_CTRL
Relative Address	0x00000218
Absolute Address	0xF8000218
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	SDIO Software Reset Control

Register SDIO_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
SDIO1_REF_RST	5	rw	0x0	SDIO 1 Reference software reset. On assertion of this reset, the Reference clock portion of the SDIO 1 subsystem will be reset. 0: No reset 1: Reference clock portion of SDIO 1 subsystem held in reset
SDIO0_REF_RST	4	rw	0x0	SDIO 0 Reference software reset. On assertion of this reset, the Reference clock portion of the SDIO 0 subsystem will be reset. 0: No reset 1: Reference clock portion of SDIO 0 subsystem held in reset.
reserved	3:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
SDIO1_CPU1X_RST	1	rw	0x0	SDIO 1 master and slave AMBA software reset. On assertion of this reset, the master and slave AMBA clock portion of the SDIO 1 subsystem will be reset. 0: No reset 1: master and slave AMBA clock portion of SDIO 1 subsystem held in reset
SDIO0_CPU1X_RST	0	rw	0x0	SDIO 0 master and slave AMBA software reset. On assertion of this reset, the master and slave AMBA clock portion of the SDIO 0 subsystem will be reset. 0: No reset 1: master and slave AMBA clock portion of SDIO 0 subsystem held in reset

Register ([slcr](#)) SPI_RST_CTRL

Name	SPI_RST_CTRL
Relative Address	0x0000021C
Absolute Address	0xF800021C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	SPI Software Reset Control

Register SPI_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:4	rw	0x0	Reserved. Writes are ignored, read data is zero.
SPI1_REF_RST	3	rw	0x0	SPI 1 Reference software reset. On assertion of this reset, the Reference clock portion of the SPI 1 subsystem will be reset. 0: No reset 1: Reference clock portion of SPI 1 subsystem held in reset
SPI0_REF_RST	2	rw	0x0	SPI 0 Reference software reset. On assertion of this reset, the Reference clock portion of the SPI 0 subsystem will be reset. 0: No reset 1: Reference clock portion of SPI 0 subsystem held in reset
SPI1_CPU1X_RST	1	rw	0x0	SPI 1 AMBA software reset. On assertion of this reset, the AMBA clock portion of the SPI 1 subsystem will be reset. 0: No reset 1: AMBA clock portion of SPI 1 subsystem held in reset
SPI0_CPU1X_RST	0	rw	0x0	SPI 0 AMBA software reset. On assertion of this reset, the AMBA clock portion of the SPI 0 subsystem will be reset. 0: No reset 1: AMBA clock portion of SPI 0 subsystem held in reset

Register ([slcr](#)) CAN_RST_CTRL

Name	CAN_RST_CTRL
Relative Address	0x00000220
Absolute Address	0xF8000220
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	CAN Software Reset Control

Register CAN_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:4	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	3	rw	0x0	Reserved. Writes will maintain value
reserved	2	rw	0x0	Reserved. Writes will maintain value
CAN1_CPU1X_RST	1	rw	0x0	CAN 1 AMBA software reset. On assertion of this reset, the AMBA clock portion of the CAN 1 subsystem will be reset. 0: No reset 1: AMBA clock portion of CAN 1 subsystem held in reset
CAN0_CPU1X_RST	0	rw	0x0	CAN 0 AMBA software reset. On assertion of this reset, the AMBA clock portion of the CAN 0 subsystem will be reset. 0: No reset 1: AMBA clock portion of CAN 0 subsystem held in reset

Register ([slcr](#)) I2C_RST_CTRL

Name	I2C_RST_CTRL
Relative Address	0x00000224
Absolute Address	0xF8000224
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	I2C Software Reset Control

Register I2C_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:2	rw	0x0	Reserved. Writes are ignored, read data is zero.

Field Name	Bits	Type	Reset Value	Description
I2C1_CPU1X_RST	1	rw	0x0	I2C 1 AMBA software reset. On assertion of this reset, the AMBA clock portion of the I2C 1 subsystem will be reset. 0: No reset 1: AMBA clock portion of I2C 1 subsystem held in reset
I2C0_CPU1X_RST	0	rw	0x0	I2C 0 AMBA software reset. On assertion of this reset, the AMBA clock portion of the I2C 0 subsystem will be reset. 0: No reset 1: AMBA clock portion of I2C 0 subsystem held in reset

Register ([slcr](#)) UART_RST_CTRL

Name	UART_RST_CTRL
Relative Address	0x00000228
Absolute Address	0xF8000228
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	UART Software Reset Control

Register UART_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:4	rw	0x0	Reserved. Writes are ignored, read data is zero.
UART1_REF_RST	3	rw	0x0	UART 1 Reference software reset. 0: deassert soft reset 1: assert soft reset
UART0_REF_RST	2	rw	0x0	UART 0 Reference software reset. 0: deassert soft reset 1: assert soft reset

Field Name	Bits	Type	Reset Value	Description
UART1_CPU1X_RST	1	rw	0x0	UART 1 AMBA software reset. On assertion of this reset, the AMBA clock portion of the UART 1 subsystem will be reset. 0: No reset 1: AMBA clock portion of UART 1 subsystem held in reset
UART0_CPU1X_RST	0	rw	0x0	UART 0 AMBA software reset. On assertion of this reset, the AMBA clock portion of the UART 0 subsystem will be reset. 0: No reset 1: AMBA clock portion of UART 0 subsystem held in reset

Register ([slcr](#)) GPIO_RST_CTRL

Name	GPIO_RST_CTRL
Relative Address	0x0000022C
Absolute Address	0xF800022C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	GPIO Software Reset Control

Register GPIO_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
GPIO_CPU1X_RST	0	rw	0x0	GPIO AMBA software reset. On assertion of this reset, the AMBA clock portion of the GPIO subsystem will be reset. 0: No reset 1: AMBA clock portion of GPIO subsystem held in reset

Register ([slcr](#)) LQSPI_RST_CTRL

Name	LQSPI_RST_CTRL
Relative Address	0x00000230
Absolute Address	0xF8000230
Width	32 bits

Access Type rw

Reset Value 0x00000000

Description Quad SPI Software Reset Control

Register LQSPI_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
QSPI_REF_RST	1	rw	0x0	Quad SPI Reference software reset. On assertion of this reset, the Reference clock portion of the QSPI subsystem will be reset. 0: No reset. 1: Reference clock portion of QSPI subsystem held in reset.
LQSPI_CPU1X_RST	0	rw	0x0	Quad SPI AMBA software reset. On assertion of this reset, the AMBA clock portion of the LQSPI subsystem will be reset. 0: No reset 1: AMBA clock portion of QSPI subsystem held in reset

Register ([slcr](#)) SMC_RST_CTRL

Name SMC_RST_CTRL

Relative Address 0x00000234

Absolute Address 0xF8000234

Width 32 bits

Access Type rw

Reset Value 0x00000000

Description SMC Software Reset Control

Register SMC_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:2	rw	0x0	Reserved. Writes are ignored, read data is zero.

Field Name	Bits	Type	Reset Value	Description
SMC_REF_RST	1	rw	0x0	SMC Reference software reset. On assertion of this reset, the Reference clock portion of the SMC subsystem will be reset. 0: No reset 1: Reference clock portion of SMC subsystem held in reset
SMC_CPU1X_RST	0	rw	0x0	SMC AMBA software reset. On assertion of this reset, the AMBA clock portion of the SMC subsystem will be reset. 0: No reset 1: AMBA clock portion of SMC subsystem held in reset

Register ([slcr](#)) OCM_RST_CTRL

Name	OCM_RST_CTRL
Relative Address	0x00000238
Absolute Address	0xF8000238
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	OCM Software Reset Control

Register OCM_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
OCM_RST	0	rw	0x0	OCM software reset. On assertion of this reset, the OCM subsystem will be reset. 0: No reset 1: OCM subsystem held in reset

Register ([slcr](#)) DEVCI_RST_CTRL

Name	DEVCI_RST_CTRL
Relative Address	0x0000023C
Absolute Address	0xF800023C
Width	32 bits
Access Type	rw
Reset Value	0x00000000

Description Device Config Interface SW Reset Control

Register DEVC_I_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
DEVC_I_CPU1X_RST	1	rw	0x0	Device Configuration AMBA software reset. On assertion of this reset, the AMBA clock portion of the Device Config subsystem will be reset. 0: No reset 1: AMBA clock portion of Device Configuration subsystem held in reset
PCAP2X_RST	0	rw	0x0	Device Configuration PCAP2X software reset. On assertion of this reset, the PCAP2X clock portion of the Device Config subsystem will be reset. 0: No reset 1: PCAP2X clock portion of Device Configuration subsystem held in reset

Register ([slcr](#)) FPGA_RST_CTRL

Name FPGA_RST_CTRL
Relative Address 0x00000240
Absolute Address 0xF8000240
Width 32 bits
Access Type rw
Reset Value 0x01F33F0F
Description FPGA Software Reset Control

Register FPGA_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:25	rw	0x0	Reserved. Writes are ignored, read data is zero.
FPGA_ACP_RST	24	rw	0x1	FPGA ACP port soft reset: 0: No reset 1: ACP AXI interface reset output asserted
FPGA_AXDS3_RST	23	rw	0x1	AXDS3AXI interface soft reset. On assertion of this reset, the AXDS3AXI interface reset output will be asserted. 0: No reset 1: AXDS3AXI interface reset output asserted

Field Name	Bits	Type	Reset Value	Description
FPGA_AXDS2_RST	22	rw	0x1	AXDS2 AXI interface soft reset. On assertion of this reset, the AXDS2 AXI interface reset output will be asserted. 0: No reset 1: AXDS2 AXI interface reset output asserted
FPGA_AXDS1_RST	21	rw	0x1	AXDS1 AXI interface soft reset. On assertion of this reset, the AXDS1 AXI interface reset output will be asserted. 0: No reset 1: AXDS1 AXI interface reset output asserted
FPGA_AXDS0_RST	20	rw	0x1	AXDS0 AXI interface soft reset. On assertion of this reset, the AXDS0 AXI interface reset output will be asserted. 0: No reset 1: AXDS0 AXI interface reset output asserted
reserved	19:18	rw	0x0	Reserved. Writes are ignored, read data is zero.
FSSW1_FPGA_RST	17	rw	0x1	General purpose FPGA slave interface 1 soft reset. On assertion of this reset, the FPGA slave interface 1 reset will be asserted. 0: No reset 1: FPGA slave interface 1 reset is asserted
FSSW0_FPGA_RST	16	rw	0x1	General purpose FPGA slave interface 0 soft reset. On assertion of this reset, the FPGA slave interface 0 reset will be asserted. 0: No reset 1: FPGA slave interface 0 reset is asserted
reserved	15:14	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	13	rw	0x1	Reserved - always write with 0
reserved	12	rw	0x1	Reserved - always write with 0
FPGA_DMA3_RST	11	rw	0x1	FPGA DMA 3 peripheral request soft reset. On assertion of this reset, the FPGA DMA 3 peripheral request reset output will be asserted. 0: No reset 1: FPGA DMA 3 peripheral request reset output asserted
FPGA_DMA2_RST	10	rw	0x1	FPGA DMA 2 peripheral request soft reset. On assertion of this reset, the FPGA DMA 2 peripheral request reset output will be asserted. 0: No reset 1: FPGA DMA 2 peripheral request reset output asserted

Field Name	Bits	Type	Reset Value	Description
FPGA_DMA1_RST	9	rw	0x1	FPGA DMA 1 peripheral request soft reset. On assertion of this reset, the FPGA DMA 1 peripheral request reset output will be asserted. 0: No reset 1: FPGA DMA 1 peripheral request reset output asserted
FPGA_DMA0_RST	8	rw	0x1	FPGA DMA 0 peripheral request soft reset. On assertion of this reset, the FPGA DMA 0 peripheral request reset output will be asserted. 0: No reset 1: FPGA DMA 0 peripheral request reset output asserted
reserved	7:4	rw	0x0	Reserved. Writes are ignored, read data is zero.
FPGA3_OUT_RST	3	rw	0x1	FPGA3 software reset. On assertion of this reset, the FPGA 3 top level reset output will be asserted. The top level reset is called FCLKRESETN[3] in software. 0: No reset 1: FPGA 3 top level reset output asserted
FPGA2_OUT_RST	2	rw	0x1	FPGA2 software reset. On assertion of this reset, the FPGA 2 top level reset output will be asserted. The top level reset is called FCLKRESETN[2] in software. 0: No reset 1: FPGA 2 top level reset output asserted
FPGA1_OUT_RST	1	rw	0x1	FPGA1 software reset. On assertion of this reset, the FPGA 1 top level reset output will be asserted. The top level reset is called FCLKRESETN[1] in software. 0: No reset 1: FPGA 1 top level reset output asserted
FPGA0_OUT_RST	0	rw	0x1	FPGA0 software reset. On assertion of this reset, the FPGA 0 top level reset output will be asserted. The top level reset is called FCLKRESETN[0] in software. 0: No reset 1: FPGA 0 top level reset output asserted

Register ([slcr](#)) A9_CPU_RST_CTRL

Name	A9_CPU_RST_CTRL
Relative Address	0x00000244
Absolute Address	0xF8000244

Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	CPU Software Reset Control

Register A9_CPU_RST_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:9	rw	0x0	Reserved. Writes are ignored, read data is zero.
PERI_RST	8	rw	0x0	CPU peripheral soft reset. 0: 1:
reserved	7:6	rw	0x0	Reserved. Writes are ignored, read data is zero.
A9_CLKSTOP1	5	rw	0x0	Clock stop if 1 for core 1
A9_CLKSTOP0	4	rw	0x0	Clock stop if 1 for core 0
reserved	3:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
A9_RST1	1	rw	0x0	CPU 1 software reset signal assertion. 0: deassert 1: assert
A9_RST0	0	rw	0x0	CPU 0 software reset signal assertion. 0: deassert 1: assert

Register ([slcr](#)) RS_AWDT_CTRL

Name	RS_AWDT_CTRL
Relative Address	0x0000024C
Absolute Address	0xF800024C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Watchdog Timer Reset Control

Register RS_AWDT_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
CTRL1	1	rw	0x0	When the APU watchdog timer 1 goes off, the SLCR can control whether the whole system is rebooted or just the CPU whose WDT went off will be reset. 0: reset Zynq 1: reset CPU
CTRL0	0	rw	0x0	When the APU watchdog timer 0 goes off, the SLCR can control whether the whole system is rebooted or just the CPU whose WDT went off will be reset. 0: reset Zynq 1: reset CPU core

Register ([slcr](#)) REBOOT_STATUS

Name	REBOOT_STATUS
Relative Address	0x00000258
Absolute Address	0xF8000258
Width	32 bits
Access Type	rw
Reset Value	0x00400000
Description	Reboot Status, persistent

Register REBOOT_STATUS Details

The Reboot Status persistent through all resets except Power-on reset.

Field Name	Bits	Type	Reset Value	Description
REBOOT_STATE	31:24	rw	0x0	General 32-bit R/W field to allow software to store information that persists through all resets except power-on reset. This field is reset by POR only. The ROM will put the last known reset reason into this register.
reserved	23	rw	0x0	Reserved.
POR	22	rw	0x1	Last reset was due to POR (power on reset), if set. This field is written by ROM code.
SRST_B	21	rw	0x0	Last reset was due to SRST_B (soft reset), if set. This field is written by ROM code.

Field Name	Bits	Type	Reset Value	Description
DBG_RST	20	rw	0x0	Last reset was due to debug system reset, if set. This field is written by ROM code.
SLC_RST	19	rw	0x0	Last reset was due to SLC soft reset, if set. This field is written by ROM code.
AWDT1_RST	18	rw	0x0	Last reset was due to APU watchdog timer 1, if set. This field is written by ROM code.
AWDT0_RST	17	rw	0x0	Last reset was due to APU watchdog timer 0, if set. This field is written by ROM code.
SWDT_RST	16	rw	0x0	Last reset was due to system watchdog timeout, if set (see watchdog status for more details). This field is written by ROM code.
BOOTROM_ERROR_CODE	15:0	rw	0x0	This field is written by the BOOTROM to describe any errors that occur during the boot process. See section 6.3.6 Debug Status for details.

Register ([slcr](#)) BOOT_MODE

Name	BOOT_MODE
Relative Address	0x0000025C
Absolute Address	0xF800025C
Width	32 bits
Access Type	mixed
Reset Value	x
Description	Boot Mode Strapping Pins

Register BOOT_MODE Details

Boot mode strapping pins are sampled when Power-on Reset deasserts. The logic levels are stored in this register. The explanation of these boot mode pin settings are explained in the boot mode section of the Zynq Technical Reference Manual.

Field Name	Bits	Type	Reset Value	Description
reserved	31:5	rw	0x0	Reserved. Writes are ignored, read data is zero.

Field Name	Bits	Type	Reset Value	Description
PLL_BYPASS	4	ro	0x0	Boot mode pins are sampled when Power-on Reset deasserts. The logic levels are stored in this register. The PLL_BYPASS pin sets the initial operating mode of all three PLL clocks (ARM, IO and DDR): 0: PLLs are enabled and their outputs are routed to the clock generators 1: PLLs are disabled and bypassed
BOOT_MODE	3:0	ro	x	Boot mode pins are sampled when Power-on Reset deasserts. The logic levels are stored in this register. The interpretation of these boot mode values are explained in the boot mode section of the Zynq Technical Reference Manual.

Register ([slcr](#)) APU_CTRL

Name	APU_CTRL
Relative Address	0x00000300
Absolute Address	0xF8000300
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	APU Control

Register APU_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:3	rw	0x0	Reserved. Writes are ignored, read data is zero.
CFGSDISABLE	2	rw	0x0	Disable write access to some system control processor registers, and some GIC registers. Set only. Once set, individual core reset cannot reset this value. This field is reset by POR only.
CP15SDISABLE	1:0	rw	0x0	Disable write access to some system control processor (CP15) registers, in each processor. Set only. Once set, individual core reset cannot reset this value. This field is reset by POR only.

Register ([slcr](#)) WDT_CLK_SEL

Name	WDT_CLK_SEL
------	-------------

Relative Address	0x00000304
Absolute Address	0xF8000304
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	APU watchdog timer clock select

Register WDT_CLK_SEL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
SEL	0	rw	0x0	System watchdog timer clock source selection: 0: CPU_1x clock 1: wdt_clk_in from PL via EMIO

Register ([slcr](#)) PSS_IDCODE

Name	PSS_IDCODE
Relative Address	0x00000530
Absolute Address	0xF8000530
Width	32 bits
Access Type	ro
Reset Value	x
Description	PS IDCODE

Register PSS_IDCODE Details

Field Name	Bits	Type	Reset Value	Description
REVISION	31:28	ro	x	Revision code
FAMILY	27:21	ro	0x1B	Family code
SUBFAMILY	20:17	ro	0x9	Subfamily code
DEVICE	16:12	ro	x	Device code 7z010 - 0x02 7z020 - 0x07 7z030 - 0x0c 7z045 - 0x11
MANUFACTURER_ID	11:1	ro	0x49	Manufacturer ID
reserved	0	ro	0x1	Reserved. Writes are ignored, read data is one.

Register ([slcr](#)) DDR_URGENT

Name	DDR_URGENT
Relative Address	0x00000600
Absolute Address	0xF8000600
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	DDR Urgent Control

Register DDR_URGENT Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	rw	0x0	Reserved
S3_ARURGENT	7	rw	0x0	Set Read port 3 prioritization.
S2_ARURGENT	6	rw	0x0	Set Read port 3 prioritization.
S1_ARURGENT	5	rw	0x0	Set Read port 2 prioritization.
S0_ARURGENT	4	rw	0x0	Set Read port 0 prioritization.
S3_AWURGENT	3	rw	0x0	Set Write port 3 prioritization.
S2_AWURGENT	2	rw	0x0	Set Write port 2 prioritization.
S1_AWURGENT	1	rw	0x0	Set Write port 1 prioritization.
S0_AWURGENT	0	rw	0x0	Set Write port 0 prioritization.

Register ([slcr](#)) DDR_CAL_START

Name	DDR_CAL_START
Relative Address	0x0000060C
Absolute Address	0xF800060C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	DDR Calibration Start Triggers

Register DDR_CAL_START Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:2	rw	0x0	Reserved. Writes are ignored, read data is zero.
START_CAL_DLL	1	wo	0x0	<p>This register creates a pulse that is first synchronised into the ddr_clk domain and then directly drives the co_gs_dll_calib input into the DDR controller. This signal is a command that indicates to the controller to issue a dll_calib to the DRAM. This signal should pulse for 1 ddr_core_clk clock cycle to request a dll_calib to be issued. This is only required if the DDR controller register reg_ddrc_dis_dll_calib is 1. If reg_ddrc_dis_dll_calib is 0, the controller will automatically issue DLL Calibs.</p> <p>0: Do nothing. 1: Start DLL calibration command. A read of this register returns zero.</p>
START_CAL_SHORT	0	wo	0x0	<p>This register creates a pulse that is first synchronized into the ddr_clk domain and then directly drives the co_gs_zq_calib_short input into the DDR controller. This is required to pulse for 1 clock to issue ZQ Calibration Short Command to the DDR. There should be a minimum of 512 clks gap between 2 ZQ Calib Short commands from the core. If DDR controller register reg_ddrc_dis_auto_zq=0, asserting co_gs_zq_calib_short is not required, as this will be done automatically. If reg_ddrc_dis_auto_zq=1, then the core logic is required to assert co_gs_zq_calib_short periodically to update DDR3 ZQ calibration.</p> <p>0: Do nothing. 1: Start ZQ calibration short command. A read of this register returns zero.</p>

Register ([slcr](#)) DDR_REF_START

Name	DDR_REF_START
Relative Address	0x00000614
Absolute Address	0xF8000614
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	DDR Refresh Start Triggers

Register DDR_REF_START Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
START_REF	0	wo	0x0	<p>This register creates a pulse that is first synchronized into the ddr_clk domain and then directly drives the co_gs_rank_refresh input into the DDR controller. This register must be used with the Virage DRAM controller register bit reg_ddrc_dis_auto_refresh.</p> <p>This signal is a command that indicates to the controller to issue a refresh to the DRAM. One bit per rank. This signal should pulse for 1 ddrc_core_clk clock cycle to request a refresh to be issued.</p> <p>0: Do nothing. 1: Start refresh.</p> <p>A read of this register returns zero.</p>

Register ([slcr](#)) DDR_CMD_STA

Name	DDR_CMD_STA
Relative Address	0x00000618
Absolute Address	0xF8000618
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	DDR Command Store Status

Register DDR_CMD_STA Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
CMD_Q_NEMPTY	0	ro	0x0	<p>DDR controller command store fill status.</p> <p>0: indicates DDRC command store is empty. 1: indicates there are commands pending in DDRC command store.</p> <p>This register is a continuous monitor of the ddrc_co_q_not_empty output from the DDR controller, which is first synchronised from ddr_clk into amba1x_clk.</p>

Register ([slcr](#)) DDR_URGENT_SEL

Name	DDR_URGENT_SEL
Relative Address	0x0000061C
Absolute Address	0xF800061C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	DDR Urgent Select

Register DDR_URGENT_SEL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	rw	0x0	Reserved. Writes are ignored, read data is zero.
S3_ARQOS_MODE	15:14	rw	0x0	<p>Selects between the AXI port s3_awqos[3], fabric signal or static register to drive the DDRC urgent bit.</p> <p>00: DDRC s3_awurgent bit is driven from the 'S3_AWURGENT' field of the DDR_URGENT_VAL register.</p> <p>01: DDRC s3_awurgent bit is driven from the s3_awqos bit.</p> <p>10: DDRC s3_awurgent bit is driven from the fabric ddr_arb[3] input.</p> <p>11: undefined</p>
S2_ARQOS_MODE	13:12	rw	0x0	<p>Selects between the AXI port s2_arqos[3], fabric signal or static register to drive the DDRC urgent bit.</p> <p>00: DDRC s2_arurgent bit is driven from the 'S2_ARURGENT' field of the DDR_URGENT_VAL register.</p> <p>01: DDRC s2_arurgent bit is driven from the s2_arqos bit.</p> <p>10: DDRC s2_arurgent bit is driven from the fabric ddr_arb[2] input.</p> <p>11: undefined</p>

Field Name	Bits	Type	Reset Value	Description
S1_ARQOS_MODE	11:10	rw	0x0	<p>Selects between the AXI port s1_arqos[3], fabric signal or static register to drive the DDRC urgent bit.</p> <p>00: DDRC s1_arurgent bit is driven from the 'S1_ARURGENT' field of the DDR_URGENT_VAL register.</p> <p>01: DDRC s1_arurgent bit is driven from the s1_arqos bit.</p> <p>10: DDRC s1_arurgent bit is driven from the fabric ddr_arb[1] input.</p> <p>11: undefined.</p>
S0_ARQOS_MODE	9:8	rw	0x0	<p>Selects between the fabric signal or static register to drive the DDRC urgent bit.</p> <p>00: DDRC s0_arurgent bit is driven from the 'S0_ARURGENT' field of the DDR_URGENT_VAL register.</p> <p>x1: undefined</p> <p>10: DDRC s0_arurgent bit is driven from the fabric ddr_arb[0] input.</p> <p>11: undefined</p>
S3_AWQOS_MODE	7:6	rw	0x0	<p>Selects between the AXI port s3_awqos[3], fabric signal or static register to drive the DDRC urgent bit.</p> <p>00: DDRC s3_awurgent bit is driven from the 'S3_AWURGENT' field of the DDR_URGENT_VAL register.</p> <p>01: DDRC s3_awurgent bit is driven from the s3_awqos bit.</p> <p>10: DDRC s3_awurgent bit is driven from the fabric ddr_arb[3] input.</p> <p>11: undefined</p>
S2_AWQOS_MODE	5:4	rw	0x0	<p>Selects between the AXI port s2_awqos[3], fabric signal or static register to drive the DDRC urgent bit.</p> <p>00: DDRC s2_awurgent bit is driven from the 'S2_AWURGENT' field of the DDR_URGENT_VAL register.</p> <p>01: DDRC s2_awurgent bit is driven from the s2_awqos bit.</p> <p>10: DDRC s2_awurgent bit is driven from the fabric ddr_arb[2] input.</p> <p>11: undefined</p>

Field Name	Bits	Type	Reset Value	Description
S1_AWQOS_MODE	3:2	rw	0x0	Selects between the AXI port s1_awqos[3], fabric signal or static register to drive the DDRC urgent bit. 00: DDRC s1_awurgent bit is driven from the 'S1_AWURGENT' field of the DDR_URGENT_VAL register. 01: DDRC s1_awurgent bit is driven from the s1_awqos bit. 10: DDRC s1_awurgent bit is driven from the fabric ddr_arb[1] input. 11: undefined
S0_AWQOS_MODE	1:0	rw	0x0	Selects between the fabric signal or static register to drive the DDRC urgent bit. 00: The DDRC s0_awurgent bit is driven from the 'S0_AWURGENT' field of the DDR_URGENT_VAL register. x1: undefined 10: The DDRC s0_awurgent bit is driven from the fabric ddr_arb[0] input. 11: undefined

Register ([slcr](#)) DDR_DFI_STATUS

Name	DDR_DFI_STATUS
Relative Address	0x00000620
Absolute Address	0xF8000620
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	DDR DFI status

Register DDR_DFI_STATUS Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	rw	0x0	Reserved. Writes are ignored, read data is zero.
DFI_CAL_ST	0	ro	0x0	This signal is intended to allow a calibration of the IOB's at a time when the DDR controller is in its calibration mode, i.e. during an idle period.

Register ([slcr](#)) MIO_PIN_00

Name	MIO_PIN_00
------	------------

Relative Address 0x00000700

Absolute Address 0xF8000700

Width 32 bits

Access Type rw

Reset Value 0x00001601

Description MIO Pin 0 Control

Register MIO_PIN_00 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Disable HSTL Input Buffer to save power when it is an output-only (IO_Type must be HSTL). 0: enable 1: disable
PULLUP	12	rw	0x1	Enables Pullup on IO Buffer pin 0: disable 1: enable
IO_Type	11:9	rw	0x3	Select the IO Buffer Type. 000: LVTTL 001: LVC MOS18 010: LVC MOS25 011, 101, 110, 111: LVC MOS33 100: HSTL
Speed	8	rw	0x0	Select IO Buffer Edge Rate, applicable when IO_Type is LVC MOS18, LVC MOS25 or LVC MOS33. 0: Slow CMOS edge 1: Fast CMOS edge
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 0 (bank 0), Input/Output others: reserved
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Chip Select 0, Output 10: NAND Flash Chip Select, Output 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: reserved

Field Name	Bits	Type	Reset Value	Description
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Quad SPI 1 chip select, Output
TRI_ENABLE	0	rw	0x1	Tri-state enable, active high. 0: disable 1: enable

Register ([slcr](#)) MIO_PIN_01

Name	MIO_PIN_01
Relative Address	0x00000704
Absolute Address	0xF8000704
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 1 Control

Register MIO_PIN_01 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 1 (bank 0), Input/Output others: reserved
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM Address Bit 25, Output 10: SRAM/NOR Chip Select 1, Output 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: reserved

Field Name	Bits	Type	Reset Value	Description
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Quad SPI 0 Chip Select, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_02

Name	MIO_PIN_02
Relative Address	0x00000708
Absolute Address	0xF8000708
Width	32 bits
Access Type	rw
Reset Value	0x00000601
Description	MIO Pin 2 Control

Register MIO_PIN_02 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x0	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 2 (bank 0), Input/Output others: reserved
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: NAND Flash ALEn, Output 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 8, Output

Field Name	Bits	Type	Reset Value	Description
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Quad SPI 0 IO Bit 0, Input/Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_03

Name	MIO_PIN_03
Relative Address	0x0000070C
Absolute Address	0xF800070C
Width	32 bits
Access Type	rw
Reset Value	0x00000601
Description	MIO Pin 3 Control

Register MIO_PIN_03 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x0	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 3 (bank 0), Input/Output others: reserved
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Data bit 0, Input/Output 10: NAND WE_B, Output 11: SDIO 1 Card Power, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 9, Output

Field Name	Bits	Type	Reset Value	Description
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Quad SPI 0 IO Bit 1, Input/Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_04

Name	MIO_PIN_04
Relative Address	0x00000710
Absolute Address	0xF8000710
Width	32 bits
Access Type	rw
Reset Value	0x00000601
Description	MIO Pin 4 Control

Register MIO_PIN_04 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x0	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 4 (bank 0), Input/Output others: reserved
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Data Bit 1, Input/Output 10: NAND Flash IO Bit 2, Input/Output 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 10, Output

Field Name	Bits	Type	Reset Value	Description
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Quad SPI 0 IO Bit 2, Input/Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_05

Name	MIO_PIN_05
Relative Address	0x00000714
Absolute Address	0xF8000714
Width	32 bits
Access Type	rw
Reset Value	0x00000601
Description	MIO Pin 5 Control

Register MIO_PIN_05 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x0	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 5 (bank 0), Input/Output others: reserved
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Data Bit 2, Input/Output 10: NAND Flash IO Bit 0, Input/Output 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 11, Output

Field Name	Bits	Type	Reset Value	Description
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Quad SPI 0 IO Bit 3, Input/Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_06

Name	MIO_PIN_06
Relative Address	0x00000718
Absolute Address	0xF8000718
Width	32 bits
Access Type	rw
Reset Value	0x00000601
Description	MIO Pin 6 Control

Register MIO_PIN_06 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x0	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 6 (bank 0), Input/Output others: reserved
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Data Bit 3, Input/Output 10: NAND Flash IO Bit 1, Input/Output 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 12, Output

Field Name	Bits	Type	Reset Value	Description
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Quad SPI 0 Clock, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_07

Name	MIO_PIN_07
Relative Address	0x0000071C
Absolute Address	0xF800071C
Width	32 bits
Access Type	rw
Reset Value	0x00000601
Description	MIO Pin 7 Control

Register MIO_PIN_07 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x0	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 7 (bank 0), Output-only others: reserved
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR OE_B, Output 10: NAND Flash CLE_B, Output 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 13, Output

Field Name	Bits	Type	Reset Value	Description
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_08

Name	MIO_PIN_08
Relative Address	0x00000720
Absolute Address	0xF8000720
Width	32 bits
Access Type	rw
Reset Value	0x00000601
Description	MIO Pin 8 Control

Register MIO_PIN_08 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x0	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 8 (bank 0), Output-only 001: CAN 1 Tx, Output 010: SRAM/NOR BLS_B, Output 011 to 110: reserved 111: UART 1 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: NAND Flash RD_B, Output 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 14, Output

Field Name	Bits	Type	Reset Value	Description
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Quad SPI Feedback Clock, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_09

Name	MIO_PIN_09
Relative Address	0x00000724
Absolute Address	0xF8000724
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 9 Control

Register MIO_PIN_09 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 9 (bank 0), Input/Output 001: CAN 1 Rx, Input 010 to 110: reserved 111: UART 1 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Data Bit 6, Input/Output 10: NAND Flash IO Bit 4, Input/Output 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 15, Output

Field Name	Bits	Type	Reset Value	Description
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Quad SPI 1 Flash Memory Clock, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_10

Name	MIO_PIN_10
Relative Address	0x00000728
Absolute Address	0xF8000728
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 10 Control

Register MIO_PIN_10 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 10 (bank 0), Input/Output 001: CAN 0 Rx, Input 010: I2C 0 Serial Clock, Input/Output 011: PJTAG TDI, Input 100: SDIO 1 IO Bit 0, Input/Output 101: SPI 1 MOSI, Input/Output 110: reserved 111: UART 0 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Data Bit 7, Input/Output 10: NAND Flash IO Bit 5, Input/Output 11: SDIO 0 Power Control, Output

Field Name	Bits	Type	Reset Value	Description
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 2, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Quad SPI 1 IO Bit 0, Input/Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_11

Name	MIO_PIN_11
Relative Address	0x0000072C
Absolute Address	0xF800072C
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 11 Control

Register MIO_PIN_11 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 11 (bank 0), Input/Output 001: CAN 0 Tx, Output 010: I2C 0 Serial Data, Input/Output 011: PJTAG TDO, Output 100: SDIO 1 Command, Input/Output 101: SPI 1 MISO, Input/Output 110: reserved 111: UART 0 TxD, Output

Field Name	Bits	Type	Reset Value	Description
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Data Bit 4, Input/Output 10: NAND Flash IO Bit 6, Input/Output 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 3, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Quad SPI 1 IO Bit 1, Input/Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_12

Name	MIO_PIN_12
Relative Address	0x00000730
Absolute Address	0xF8000730
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 12 Control

Register MIO_PIN_12 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]

Field Name	Bits	Type	Reset Value	Description
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 12 (bank 0), Input/Output 001: CAN 1 Tx, Output 010: I2C 1 Serial Clock, Input/Output 011: PJTAG TCK, Input 100: SDIO 1 Clock, Input/Output 101: SPI 1 Serial Clock, Input/Output 110: reserved 111: UART 1 Tx/D, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Wait, Input 10: NAND Flash IO Bit 7, Input/Output 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Clock, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Quad SPI 1 IO Bit 2, Input/Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_13

Name	MIO_PIN_13
Relative Address	0x00000734
Absolute Address	0xF8000734
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 13 Control

Register MIO_PIN_13 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]

Field Name	Bits	Type	Reset Value	Description
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 13 (bank 0), Input/Output 001: CAN 1 Rx, Input 010: I2C 1 Serial Data, Input/Output 011: PJTAG TMS, Input 100: SDIO 1 IO Bit 1, Input/Output 101: SPI 1 Slave Select 0, Input/Output 110: reserved 111: UART 1 Rx/D, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Data Bit 5, Input/Output 10: NAND Flash IO Bit 3, Input/Output 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Control Signal, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Quad SPI 1 IO Bit 3, Input/Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_14

Name	MIO_PIN_14
Relative Address	0x00000738
Absolute Address	0xF8000738
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 14 Control

Register MIO_PIN_14 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 14 (bank 0), Input/Output 001: CAN 0 Rx, Input 010: I2C 0 Serial Clock, Input/Output 011: SWDT Clock, Input 100: SDIO 1 IO Bit 2, Input/Output 101: SPI 1 slave select 1, Output 110: reserved 111: UART 0 Rx/D, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: NAND Flash Busy, Input 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 0, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1= Not Used
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_15

Name	MIO_PIN_15
Relative Address	0x0000073C
Absolute Address	0xF800073C
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 15 Control

Register MIO_PIN_15 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 15 (bank 0), Input/Output 001: CAN 0 Tx, Output 010: I2C 0 Serial Data, Input/Output 011: SWDT Reset, Output 100: SDIO 1 IO Bit 3, Input/Output 101: SPI 1 Slave Select 2, Output 110: reserved 111: UART 0 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 0, Output 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 1, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1= Not Used
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_16

Name	MIO_PIN_16
Relative Address	0x00000740
Absolute Address	0xF8000740
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 16 Control

Register MIO_PIN_16 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 16 (bank 0), Input/Output 001: CAN 1 Tx, Output 010: I2C 1 Serial Clock, Input/Output 011: reserved 100: SDIO 0 Clock, Input/Output 101: SPI 0 Serial Clock, Input/Output 110: TTC 1 Wave, Output 111: UART 1 Tx/D, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 1, Output 10: NAND Flash IO Bit 8, Input/Output 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 4, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 0 RGMII Tx Clock, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_17

Name	MIO_PIN_17
Relative Address	0x00000744
Absolute Address	0xF8000744
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 17 Control

Register MIO_PIN_17 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 17 (bank 0), Input/Output 001: CAN 1 Rx, Input 010: I2C 1 Serial Data, Input/Output 011: reserved 100: SDIO 0 Command, Input/Output 101: SPI 0 MISO, Input/Output 110 TTC 1 Clock, Input 111: UART 1 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 2, Output 10: NAND Flash IO Bit 9, Input/Output 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 5, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 0 RGMII TxD Bit 0, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_18

Name	MIO_PIN_18
Relative Address	0x00000748
Absolute Address	0xF8000748
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 18 Control

Register MIO_PIN_18 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 18 (bank 0), Input/Output 001: CAN 0 Rx, Input 010: I2C 0 Serial Clock, Input/Output 011: reserved 100: SDIO 0 IO Bit 0, Input/Output 101: SPI 0 Slave Select 0, Input/Output 110: TTC 0 Wave, Output 111: UART 0 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 3, Output 10: NAND Flash IO Bit 10, Input/Output 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 6, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 0 RGMII TxD Bit 1, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_19

Name	MIO_PIN_19
Relative Address	0x0000074C
Absolute Address	0xF800074C
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 19 Control

Register MIO_PIN_19 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 19 (bank 0), Input/Output 001: CAN 0 Tx, Output 010: I2C 0 Serial Data, Input/Output 011: reserved 100: SDIO 0 IO Bit 1, Input/Output 101: SPI 0 Slave Select 1, Output 110: TTC 0 Clock, Input 111: UART 0 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 4, Output 10: NAND Flash IO Bit 11, Input/Output 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 7, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 0 RGMII TxD Bit 2, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_20

Name	MIO_PIN_20
Relative Address	0x00000750
Absolute Address	0xF8000750
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 20 Control

Register MIO_PIN_20 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 20 (bank 0), Input/Output 001: CAN 1 Tx, Output 010: I2C 1 Serial Clock, Input/Output 011: reserved 100: SDIO 0 IO Bit 2, Input/Output 101: SPI 0 Slave Select 2, Output 110: reserved 111: UART 1 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 5, Output 10: NAND Flash IO Bit 12, Input/Output 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: reserved
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 0 RGMII TxD Bit 3, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_21

Name	MIO_PIN_21
Relative Address	0x00000754
Absolute Address	0xF8000754
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 21 Control

Register MIO_PIN_21 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 21 (bank 0), Input/Output 001: CAN 1 Rx, Input 010: I2C 1 Serial Data, Input/Output 011: reserved 100: SDIO 0 IO Bit 3, Input/Output 101: SPI 0 MOSI, Input/Output 110: reserved 111: UART 1 Rx/D, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 6, Output 10: NAND Flash IO Bit 13, Input/Output 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: reserved
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 0 RGMII Tx Control, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_22

Name	MIO_PIN_22
Relative Address	0x00000758
Absolute Address	0xF8000758
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 22 Control

Register MIO_PIN_22 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 22 (bank 0), Input/Output 001: CAN 0 Rx, Input 010: I2C 0 Serial Clock, Input/Output 011: PJTAG TDI, Input 100: SDIO 1 IO Bit 0, Input/Output 101: SPI 1 MOSI, Input/Output 110: reserved 111: UART 0 Rx/D, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 7, Output 10: NAND Flash IO Bit 14, Input/Output 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 2, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 0 RGMII Rx Clock, Input
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_23

Name	MIO_PIN_23
Relative Address	0x0000075C
Absolute Address	0xF800075C
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 23 Control

Register MIO_PIN_23 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 23 (bank 0), Input/Output 001: CAN 0 Tx, Output 010: I2C 0 Serial Data, Input/Output 011: PJTAG TDO, Output 100: SDIO 1 Command, Input/Output 101: SPI 1 MISO, Input/Output 110: reserved 111: UART 0 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 8, Output 10: NAND Flash IO Bit 15, Input/Output 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 3, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 0 RGMII RxD 0, Input
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_24

Name	MIO_PIN_24
Relative Address	0x00000760
Absolute Address	0xF8000760
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 24 Control

Register MIO_PIN_24 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 24 (bank 0), Input/Output 001: CAN 1 Tx, Output 010: I2C 1 Serial Clock, Input/Output 011: PJTAG TCK, Input 100: SDIO 1 Clock, Input/Output 101: SPI 1 Serial Clock, Input/Output 110: reserved 111: UART 1 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 9, Output 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Clock output, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 0 RGMII RxD Bit 1, Input
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_25

Name	MIO_PIN_25
Relative Address	0x00000764
Absolute Address	0xF8000764
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 25 Control

Register MIO_PIN_25 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 25 (bank 0), Input/Output 001: CAN 1 Rx, Input 010: I2C 1 Serial Data, Input/Output 011: PJTAG TMS, Input 100: SDIO 1 IO Bit 1, Input/Output 101: SPI 1 Slave Select 0, Input/Output 110: reserved 111: UART 1 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 10, Output 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Control Signal, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 0 RGMII RxD Bit2, Input
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_26

Name	MIO_PIN_26
Relative Address	0x00000768
Absolute Address	0xF8000768
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 26 Control

Register MIO_PIN_26 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 26 (bank 0), Input/Output 001: CAN 0 Rx, Input 010: I2C 0 Serial Clock, Input/Output 011: SWDT Clock, Input 100: SDIO 1 IO Bit 2, Input/Output 101: SPI 1 Slave Select 1, Output 110: reserved 111: UART 0 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 11, Output 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 0, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 0 RGMII RxD Bit 3, Input
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_27

Name	MIO_PIN_27
Relative Address	0x0000076C
Absolute Address	0xF800076C
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 27 Control

Register MIO_PIN_27 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 27 (bank 0), Input/Output 001: CAN 0 Tx, Output 010: I2C 0 Serial Data, Input/Output 011: SWDT Reset, Output 100: SDIO 1 IO Bit 3, Input/Output 101: SPI 1 Slave Select 2, Output 110: reserved 111: UART 0 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 12, Output 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: Trace Port Data Bit 1, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 0 RGMII Rx Control, Input
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_28

Name	MIO_PIN_28
Relative Address	0x00000770
Absolute Address	0xF8000770
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 28 Control

Register MIO_PIN_28 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 28 (bank 0), Input/Output 001: CAN 1 Tx, Output 010: I2C 1 Serial Clock, Input/Output 011: reserved 100: SDIO 0 Clock, Input/Output 101: SPI 0 Serial Clock, Input/Output 110: TTC 1 Wave, Output 111: UART 1 Tx/D, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 13, Output 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 0 ULPI Data Bit 4, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 1 RGMII Tx Clock, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_29

Name	MIO_PIN_29
Relative Address	0x00000774
Absolute Address	0xF8000774
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 29 Control

Register MIO_PIN_29 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 29 (bank 0), Input/Output 001: CAN 1 Rx, Input 010: I2C 1 Serial Data, Input/Output 011: reserved 100: SDIO 0 Command, Input/Output 101: SPI 0 MISO, Input/Output 110: TTC 1 Clock, Input 111: UART 1 Rx/D, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 14, Output 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 0 ULPI Direction, Input
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 1 RGMII Tx/D Bit 0, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_30

Name	MIO_PIN_30
Relative Address	0x00000778
Absolute Address	0xF8000778
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 30 Control

Register MIO_PIN_30 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 30 (bank 0), Input/Output 001: CAN 0 Rx, Input 010: I2C 0 Serial Clock, Input/Output 011: reserved 100: SDIO 0 IO Bit 0, Input/Output 101: SPI 0 Slave Select 0, Input/Output 110: TTC 0 Wave, Output 111: UART 0 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 15, Output 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 0 ULPI Stop, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 1 RGMII TxD Bit 1, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_31

Name	MIO_PIN_31
Relative Address	0x0000077C
Absolute Address	0xF800077C
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 31 Control

Register MIO_PIN_31 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 31 (bank 0), Input/Output 001: CAN 0 Tx, Output 010: I2C 0 Serial Data, Input/Output 011: reserved 100: SDIO 0 IO Bit 1, Input/Output 101: SPI 0 Slave Select 1, Output 110: TTC 0 Clock, Input 111: UART 0 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 16, Output 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 0 ULPI Next, Input
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 1 RGMII TxD Bit 2, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_32

Name	MIO_PIN_32
Relative Address	0x00000780
Absolute Address	0xF8000780
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 32 Control

Register MIO_PIN_32 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 32 (bank 1), Input/Output 001: CAN 1 Tx, Output 010: I2C 1 Serial Clock, Input/Output 011: reserved 100: SDIO 0 IO Bit 2, Input/Output 101: SPI 0 Slave Select 2, Output 110: reserved 111: UART 1 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 17, Output 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 0 ULPI Data Bit 0, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 1 RGMII TxD Bit 3, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_33

Name	MIO_PIN_33
Relative Address	0x00000784
Absolute Address	0xF8000784
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 33 Control

Register MIO_PIN_33 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 33 (Bank 1), Input/Output 001: CAN 1 Rx, Input 010: I2C 1 Serial Data, Input/Output 011: reserved 100: SDIO 0 IO Bit 3, Input/Output 101: SPI 0 MOSI, Input/Output 110: reserved 111: UART 1 Rx/D, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 18, Output 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 0 ULPI Data Bit 1, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 1 RGMII Tx Control, Output
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_34

Name	MIO_PIN_34
Relative Address	0x00000788
Absolute Address	0xF8000788
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 34 Control

Register MIO_PIN_34 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 34 (bank 1), Input/Output 001: CAN 0 Rx, Input 010: I2C 0 Serial Clock, Input/Output 011: PJTAG TDI, Input 100: SDIO 1 IO Bit 0, Input/Output 110: reserved 111: UART 0 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 19, Output 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 0 ULPI Data Bit 2, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 1 RGMII Rx Clock, Input
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_35

Name	MIO_PIN_35
Relative Address	0x0000078C
Absolute Address	0xF800078C
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 35 Control

Register MIO_PIN_35 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 35 (bank 1), Input/Output 001: CAN 0 Tx, Output 010: I2C 0 Serial Data, Input/Output 011: PJTAG TDO, Output 100: SDIO 1 Command, Input/Output 101: SPI 1 MISO, Input/Output 110: reserved 111: UART 0 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 20, Output 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 0 ULPI Data Bit 3, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 1 RGMII RxD data Bit 0, Input
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_36

Name	MIO_PIN_36
Relative Address	0x00000790
Absolute Address	0xF8000790
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 36 Control

Register MIO_PIN_36 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 36 (bank 1), Input/Output 001: CAN 1 Tx, Output 010: I2C 1 Serial Clock, Input/Output 011: PJTAG TCK, Input 100: SDIO 1 Clock, Input/Output 101: SPI 1 Clock, Input/Output 110: reserved 111: UART 1 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 21, Output 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 0 ULPI Clock, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 1 RGMII Data Bit 1
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_37

Name	MIO_PIN_37
Relative Address	0x00000794
Absolute Address	0xF8000794
Width	32 bits
Access Type	rw
Reset Value	0x00001601

Description MIO Pin 37 Control

Register MIO_PIN_37 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 37 (bank 1), Input/Output 001: CAN 1 Rx, Input 010: I2C 1 Serial Data, Input/Output 011: PJTAG TMS, Input 100: SDIO 1 IO Bit 1, Input/Output 101: SPI 1 Slave Select 0, Input/Output 110: reserved 111: UART 1 Rx/D, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 22, Output 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 0 ULPI Data Bit 5, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 1 RGMII Rx/D Data Bit 2, Input
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_38

Name MIO_PIN_38
Relative Address 0x00000798
Absolute Address 0xF8000798
Width 32 bits
Access Type rw

Reset Value 0x00001601
Description MIO Pin 38 Control

Register MIO_PIN_38 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 38 (bank 1), Input/Output 001: CAN 0 Rx, Input 010: I2C 0 Serial Clock, Input/Output 011: SWDT Clock, Input 100: SDIO 1 IO Bit 2, Input/Output 101: SPI 1 Slave Select 1, Output 110: reserved 111: UART 0 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 23, Output 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 0 ULPI Data Bit 6, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 1 RGMII RxD Data Bit 3, Input
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_39

Name MIO_PIN_39
Relative Address 0x0000079C
Absolute Address 0xF800079C
Width 32 bits

Access Type rw
Reset Value 0x00001601
Description MIO Pin 39 Control

Register MIO_PIN_39 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 39 (bank 1), Input/Output 001: CAN 0 Tx, Output 010: I2C 0 Serial Data, Input/Output 011: SWDT Reset, Output 100: SDIO 1 IO Bit 3, Input/Output 101: SPI 1 Slave Select 2, Output 110: reserved 111: UART 0 Tx, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: SRAM/NOR Address Bit 24, Output 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 0 ULPI Data Bit 7, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: Ethernet 1 RGMII Rx Control, Input
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_40

Name MIO_PIN_40
Relative Address 0x000007A0
Absolute Address 0xF80007A0

Width 32 bits

Access Type rw

Reset Value 0x00001601

Description MIO Pin 40 Control

Register MIO_PIN_40 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 40 (bank 1), Input/Output 001: CAN 1 Tx, Output 010: I2C 1 Serial Clock, Input/Output 011: reserved 100: SDIO 0 Clock, Input/Output 101: SPI 0 Serial Clock, Input/Output 110: TTC 1 Wave, Output 111: UART 1 Tx/D, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 1 ULPI Data Bit 4, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_41

Name MIO_PIN_41

Relative Address 0x000007A4

Absolute Address 0xF80007A4
 Width 32 bits
 Access Type rw
 Reset Value 0x00001601
 Description MIO Pin 41 Control

Register MIO_PIN_41 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 41 (bank 1), Input/Output 001: CAN 1 Rx, Input 010: I2C 1 Serial Data, Input/Output 011: reserved 100: SDIO 0 Command, Input/Output 101: SPI 0 MISO, Input/Output 110: TTC 1 Clock, Input 111: UART 1 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 1 ULPI Direction, Input
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_42

Name MIO_PIN_42

Relative Address	0x000007A8
Absolute Address	0xF80007A8
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 42 Control

Register MIO_PIN_42 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 42 (bank 1), Input/Output 001: CAN 0 Rx, Input 010: I2C 0 Serial Clock, Input/Output 011: reserved 100: SDIO 0 IO Bit 0, Input/Output 101: SPI 0 Slave Select 0, Input/Output 110: TTC 0 Wave, Output 111: UART 0 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 1 ULPI Stop, Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1= Not Used
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_43

Name	MIO_PIN_43
Relative Address	0x000007AC
Absolute Address	0xF80007AC
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 43 Control

Register MIO_PIN_43 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 43 (bank 1), Input/Output 001: CAN 0 Tx, Output 010: I2C 0 Serial Data, Input/Output 011: reserved 100: SDIO 0 IO Bit 1, Input/Output 101: SPI 0 Slave Select 1, Output 110: TTC 0 Clock, Input 111: UART 0 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 1 ULPI Next, Input
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_44

Name	MIO_PIN_44
Relative Address	0x000007B0
Absolute Address	0xF80007B0
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 44 Control

Register MIO_PIN_44 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 44 (bank 1), Input/Output 001: CAN 1 Tx, Output 010: I2C 1 Serial Clock, Input/Output 011: reserved 100: SDIO 0 IO Bit 2, Input/Output 101: SPI 0 Slave Select 2, Output 110: reserved 111: UART 1 Tx/D, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 1 ULPI Data Bit 0, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_45

Name	MIO_PIN_45
Relative Address	0x000007B4
Absolute Address	0xF80007B4
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 45 Control

Register MIO_PIN_45 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 45 (bank 1), Input/Output 001: CAN 1 Rx, Input 010: I2C 1 Serial Data, Input/Output 011: reserved 100: SDIO 0 IO Bit 3, Input/Output 101: SPI 0 MOSI, Input/Output 110: reserved 111: UART 1 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 1 ULPI Data Bit 1, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_46

Name	MIO_PIN_46
Relative Address	0x000007B8
Absolute Address	0xF80007B8
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 46 Control

Register MIO_PIN_46 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 46 (bank 1), Input/Output 001: CAN 0 Rx, Input 010: I2C 0 Serial Clock, Input/Output 011: PJTAG TDI, Input 100: SDIO 1 IO Bit 0, Input/Output 101: SPI 1 MOSI, Input/Output 110: reserved 111: UART 0 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 1 ULPI Data Bit 2, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_47

Name	MIO_PIN_47
Relative Address	0x000007BC
Absolute Address	0xF80007BC
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 47 Control

Register MIO_PIN_47 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 47 (bank 1), Input/Output 001: CAN 0 Tx, Output 010: I2C 0 Serial Data, Input/Output 011: PJTAG TDO, Output 100: SDIO 1 Command, Input/Output 101: SPI 1 MISO, Input/Output 110: reserved 111: UART 0 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 1 ULPI Data Bit 3, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_48

Name	MIO_PIN_48
Relative Address	0x000007C0
Absolute Address	0xF80007C0
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 48 Control

Register MIO_PIN_48 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 48 (bank 1), Input/Output 001: CAN 1 Tx, Output 010: I2C 1 Serial Clock, Input/Output 011: PJTAG TCK, Input 100: SDIO 1 Clock, Input/Output 101: SPI 1 Serial Clock, Input/Output 110: reserved 111: UART 1 Tx/D, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 1 ULPI Clock, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_49

Name	MIO_PIN_49
Relative Address	0x000007C4
Absolute Address	0xF80007C4
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 49 Control

Register MIO_PIN_49 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 49 (bank 1), Input/Output 001: CAN 1 Rx, Input 010: I2C 1 Serial Data, Input/Output 011: PJTAG TMS, Input 100: SDIO 1 IO Bit 1, Input/Output 101: SPI 1 Select 0, Input/Output 110: reserved 111: UART 1 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 1 ULPI Data Bit 5, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_50

Name	MIO_PIN_50
Relative Address	0x000007C8
Absolute Address	0xF80007C8
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 50 Control

Register MIO_PIN_50 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 50 (bank 1), Input/Output 001: CAN 0 Rx, Input 010: I2C 0 Serial Clock, Input/Output 011: SWDT Clock, Input 100: SDIO 1 IO Bit 2, Input/Output 101: SPI 1 Slave Select 1, Output 110: reserved 111: UART 0 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 1 ULPI Data Bit 6, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_51

Name	MIO_PIN_51
Relative Address	0x000007CC
Absolute Address	0xF80007CC
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 51 Control

Register MIO_PIN_51 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 51 (bank 1), Input/Output 001: CAN 0 Tx, Output 010: I2C 0 Serial Data, Input/Output 011: SWDT Reset, Output 100: SDIO 1 IO Bit 3, Input/Output 101: SPI 1 Slave Select 2, Output 110: reserved 111: UART 0 TxD, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: USB 1 ULPI Data Bit 7, Input/Output
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_52

Name	MIO_PIN_52
Relative Address	0x000007D0
Absolute Address	0xF80007D0
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 52 Control

Register MIO_PIN_52 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 52 (bank 1), Input/Output 001: CAN 1 Tx, Output 010: I2C 1 Serial Clock, Input/Output 011: SWDT Clock, Input 100: MDIO 0 Clock, Output 101: MDIO 1 Clock, Output 110: reserved 111: UART 1 Tx/D, Output
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 0 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: reserved
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_PIN_53

Name	MIO_PIN_53
Relative Address	0x000007D4
Absolute Address	0xF80007D4
Width	32 bits
Access Type	rw
Reset Value	0x00001601
Description	MIO Pin 53 Control

Register MIO_PIN_53 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	rw	0x0	reserved
DisableRcvr	13	rw	0x0	Operates the same as MIO_PIN_00[DisableRcvr]
PULLUP	12	rw	0x1	Operates the same as MIO_PIN_00[PULL_UP]
IO_Type	11:9	rw	0x3	Operates the same as MIO_PIN_00[IO_Type]
Speed	8	rw	0x0	Operates the same as MIO_PIN_00[Speed]
L3_SEL	7:5	rw	0x0	Level 3 Mux Select 000: GPIO 53 (bank 1), Input/Output 001: CAN 1 Rx, Input 010: I2C 1 Serial Data, Input/Output 011: SWDT Reset, Output 100: MDIO 0 Data, Input/Output 101: MDIO 1 Data, Input/Output 110: reserved 111: UART 1 RxD, Input
L2_SEL	4:3	rw	0x0	Level 2 Mux Select 00: Level 3 Mux 01: reserved 10: reserved 11: SDIO 1 Power Control, Output
L1_SEL	2	rw	0x0	Level 1 Mux Select 0: Level 2 Mux 1: reserved
L0_SEL	1	rw	0x0	Level 0 Mux Select 0: Level 1 Mux 1: reserved
TRI_ENABLE	0	rw	0x1	Operates the same as MIO_PIN_00[TRI_ENABLE]

Register ([slcr](#)) MIO_FMIO_GEM_SEL

Name	MIO_FMIO_GEM_SEL
Relative Address	0x00000800
Absolute Address	0xF8000800
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Ethernet Routing Select

Register MIO_FMIO_GEM_SEL Details

Select between RGMII to/from MIO and GMII to/from EMIO.

Field Name	Bits	Type	Reset Value	Description
reserved	31:2	rw	0x0	Reserved
GEM1_IF_SELECT	1	rw	0x0	Ethernet Controller 1 Interface selection 0: RGMII to/from MIO 1: GMII to/from EMIO The setting for this register generates the gem0_rx_sel signals to the Ethernet core wrappers within the IOP.
GEM0_IF_SELECT	0	rw	0x0	Ethernet Controller 0 Interface selection. 0: RGMII to/from MIO 1: GMII to/from EMIO The setting for this register generates the gem0_rx_sel signals to the Ethernet core wrappers within the IOP.

Register ([slcr](#)) MIO_LOOPBACK

Name	MIO_LOOPBACK
Relative Address	0x00000804
Absolute Address	0xF8000804
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Loopback function within MIO

Register MIO_LOOPBACK Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:4	rw	0x0	reserved
I2C0_LOOP_I2C1	3	rw	0x0	I2C Loopback Control. 0 = Connect I2C inputs according to MIO mapping. 1 = Loop I2C 0 outputs to I2C 1 inputs, and I2C 1 outputs to I2C 0 inputs.
CAN0_LOOP_CAN1	2	rw	0x0	CAN Loopback Control. 0 = Connect CAN inputs according to MIO mapping. 1 = Loop CAN 0 Tx to CAN 1 Rx, and CAN 1 Tx to CAN 0 Rx.
UA0_LOOP_UA1	1	rw	0x0	UART Loopback Control. 0 = Connect UART inputs according to MIO mapping. 1 = Loop UART 0 outputs to UART 1 inputs, and UART 1 outputs to UART 0 inputs. RXD/TXD cross-connected. RTS/CTS cross-connected. DSR, DTR, DCD and RI not used.
SPI0_LOOP_SPI1	0	rw	0x0	SPI Loopback Control. 0 = Connect SPI inputs according to MIO mapping. 1 = Loop SPI 0 outputs to SPI 1 inputs, and SPI 1 outputs to SPI 0 inputs. The other SPI core will appear on the LS Slave Select.

Register ([slcr](#)) MIO_MST_TRI0

Name	MIO_MST_TRI0
Relative Address	0x0000080C
Absolute Address	0xF800080C
Width	32 bits
Access Type	rw
Reset Value	0xFFFFFFFF
Description	MIO pin Tri-state Enables, 31:0

Register MIO_MST_TRI0 Details

Parallel access to the master tri-state enables for MIO pins

Field Name	Bits	Type	Reset Value	Description
PIN_31_TRI	31	rw	0x1	Master Tri-state Enable for pin 31, active high
PIN_30_TRI	30	rw	0x1	Master Tri-state Enable for pin 30, active high
PIN_29_TRI	29	rw	0x1	Master Tri-state Enable for pin 29, active high
PIN_28_TRI	28	rw	0x1	Master Tri-state Enable for pin 28, active high
PIN_27_TRI	27	rw	0x1	Master Tri-state Enable for pin 27, active high
PIN_26_TRI	26	rw	0x1	Master Tri-state Enable for pin 26, active high
PIN_25_TRI	25	rw	0x1	Master Tri-state Enable for pin 25, active high
PIN_24_TRI	24	rw	0x1	Master Tri-state Enable for pin 24, active high
PIN_23_TRI	23	rw	0x1	Master Tri-state Enable for pin 23, active high
PIN_22_TRI	22	rw	0x1	Master Tri-state Enable for pin 22, active high
PIN_21_TRI	21	rw	0x1	Master Tri-state Enable for pin 21, active high
PIN_20_TRI	20	rw	0x1	Master Tri-state Enable for pin 20, active high
PIN_19_TRI	19	rw	0x1	Master Tri-state Enable for pin 19, active high
PIN_18_TRI	18	rw	0x1	Master Tri-state Enable for pin 18, active high
PIN_17_TRI	17	rw	0x1	Master Tri-state Enable for pin 17, active high
PIN_16_TRI	16	rw	0x1	Master Tri-state Enable for pin 16, active high
PIN_15_TRI	15	rw	0x1	Master Tri-state Enable for pin 15, active high
PIN_14_TRI	14	rw	0x1	Master Tri-state Enable for pin 14, active high
PIN_13_TRI	13	rw	0x1	Master Tri-state Enable for pin 13, active high
PIN_12_TRI	12	rw	0x1	Master Tri-state Enable for pin 12, active high
PIN_11_TRI	11	rw	0x1	Master Tri-state Enable for pin 11, active high
PIN_10_TRI	10	rw	0x1	Master Tri-state Enable for pin 10, active high
PIN_09_TRI	9	rw	0x1	Master Tri-state Enable for pin 9, active high
PIN_08_TRI	8	rw	0x1	Master Tri-state Enable for pin 8, active high
PIN_07_TRI	7	rw	0x1	Master Tri-state Enable for pin 7, active high
PIN_06_TRI	6	rw	0x1	Master Tri-state Enable for pin 6, active high
PIN_05_TRI	5	rw	0x1	Master Tri-state Enable for pin 5, active high
PIN_04_TRI	4	rw	0x1	Master Tri-state Enable for pin 4, active high
PIN_03_TRI	3	rw	0x1	Master Tri-state Enable for pin 3, active high
PIN_02_TRI	2	rw	0x1	Master Tri-state Enable for pin 2, active high
PIN_01_TRI	1	rw	0x1	Master Tri-state Enable for pin 1, active high
PIN_00_TRI	0	rw	0x1	Master Tri-state Enable for pin 0, active high

Register ([slcr](#)) MIO_MST_TRI1

Name	MIO_MST_TRI1
Relative Address	0x00000810
Absolute Address	0xF8000810
Width	32 bits
Access Type	rw
Reset Value	0x003FFFFFFF
Description	MIO pin Tri-state Enables, 53:32

Register MIO_MST_TRI1 Details

Parallel access to the master tri-state enables for MIO pins

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rw	0x0	reserved
PIN_53_TRI	21	rw	0x1	Master Tri-state Enable for pin 53, active high
PIN_52_TRI	20	rw	0x1	Master Tri-state Enable for pin 52, active high
PIN_51_TRI	19	rw	0x1	Master Tri-state Enable for pin 51, active high
PIN_50_TRI	18	rw	0x1	Master Tri-state Enable for pin 50, active high
PIN_49_TRI	17	rw	0x1	Master Tri-state Enable for pin 49, active high
PIN_48_TRI	16	rw	0x1	Master Tri-state Enable for pin 48, active high
PIN_47_TRI	15	rw	0x1	Master Tri-state Enable for pin 47, active high
PIN_46_TRI	14	rw	0x1	Master Tri-state Enable for pin 46, active high
PIN_45_TRI	13	rw	0x1	Master Tri-state Enable for pin 45, active high
PIN_44_TRI	12	rw	0x1	Master Tri-state Enable for pin 44, active high
PIN_43_TRI	11	rw	0x1	Master Tri-state Enable for pin 43, active high
PIN_42_TRI	10	rw	0x1	Master Tri-state Enable for pin 42, active high
PIN_41_TRI	9	rw	0x1	Master Tri-state Enable for pin 41, active high
PIN_40_TRI	8	rw	0x1	Master Tri-state Enable for pin 40, active high
PIN_39_TRI	7	rw	0x1	Master Tri-state Enable for pin 39, active high
PIN_38_TRI	6	rw	0x1	Master Tri-state Enable for pin 38, active high
PIN_37_TRI	5	rw	0x1	Master Tri-state Enable for pin 37, active high
PIN_36_TRI	4	rw	0x1	Master Tri-state Enable for pin 36, active high
PIN_35_TRI	3	rw	0x1	Master Tri-state Enable for pin 35, active high
PIN_34_TRI	2	rw	0x1	Master Tri-state Enable for pin 34, active high
PIN_33_TRI	1	rw	0x1	Master Tri-state Enable for pin 33, active high
PIN_32_TRI	0	rw	0x1	Master Tri-state Enable for pin 32, active high

Register ([slcr](#)) SD0_WP_CD_SEL

Name	SD0_WP_CD_SEL
Relative Address	0x00000830
Absolute Address	0xF8000830
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	SDIO 0 WP CD select

Register SD0_WP_CD_SEL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rw	0x0	reserved
SDIO0_CD_SEL	21:16	rw	0x0	SDIO 0 CD Select. Values 53:0 select MIO input (any pin except bits 7 and 8) Values 63:54 select EMIO input
reserved	15:6	rw	0x0	reserved
SDIO0_WP_SEL	5:0	rw	0x0	SDIO 0 WP Select. Values 53:0 select MIO input (any pin except 7 and 8) Values 63:54 select EMIO input

Register ([slcr](#)) SD1_WP_CD_SEL

Name	SD1_WP_CD_SEL
Relative Address	0x00000834
Absolute Address	0xF8000834
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	SDIO 1 WP CD select

Register SD1_WP_CD_SEL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:22	rw	0x0	reserved
SDIO1_CD_SEL	21:16	rw	0x0	SDIO 1 CD Select. Values 53:0 select MIO input (any pin except bits 7 and 8) Values 63:54 select EMIO input
reserved	15:6	rw	0x0	reserved
SDIO1_WP_SEL	5:0	rw	0x0	SDIO 1 WP Select. Values 53:0 select MIO input (any pin except 7 and 8) Values 63:54 select EMIO input

Register ([slcr](#)) LVL_SHFTR_EN

Name	LVL_SHFTR_EN
Relative Address	0x00000900
Absolute Address	0xF8000900
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Level Shifters Enable

Register LVL_SHFTR_EN Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:5	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	4	rw	0x0	Reserved. Do not modify.
USER_LVL_INP_EN_0	3	rw	0x0	Level shifter enable for PS-PL input signals
USER_LVL_OUT_EN_0	2	rw	0x0	Level shifter enable for PS-PL output signals
USER_LVL_INP_EN_1	1	rw	0x0	Level shifter enable for PS-PL input signals
USER_LVL_OUT_EN_1	0	rw	0x0	Level shifter enable for PS-PL output signals

Register ([slcr](#)) OCM_CFG

Name	OCM_CFG
Relative Address	0x00000910

Absolute Address	0xF8000910
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	OCM Address Mapping

Register OCM_CFG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:5	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	4	rw	0x0	Reserved. Do not modify.
RAM_HI	3:0	rw	0x0	Maps the OCM RAM (in 64 KB sections) to the high or low address space: 0: low address. 1: high address. RAM_HI [0] is first 64 KB RAM_HI [1] is second 64 KB RAM_HI [2] is third 64 KB RAM_HI [3] is fourth 64 KB Refer to the OCM chapter for more details.

Register ([slcr](#)) GPIOB_CTRL

Name	GPIOB_CTRL
Relative Address	0x00000B00
Absolute Address	0xF8000B00
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	PS IO Buffer Control

Register GPIOB_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:12	rw	0x0	Reserved. Writes are ignored, read data is zero.
VREF_SW_EN	11	rw	0x0	Enables the VREF switch 0: internal 1: external
reserved	10	rw	0x0	Reserved. Do not modify.
reserved	9	rw	0x0	Reserved. Do not modify.

Field Name	Bits	Type	Reset Value	Description
reserved	8	rw	0x0	Reserved. Do not modify.
reserved	7	rw	0x0	Reserved. Do not modify.
VREF_SEL	6:4	rw	0x0	Specifies GP000000 - VREF = test mode. 001 - VREF = test mode - 090. 010 - VREF = test mode - 108. 100 - VREF = test mode - 125.
reserved	3	rw	0x0	Reserved. Do not modify.
reserved	2	rw	0x0	Reserved. Do not modify.
reserved	1	rw	0x0	Reserved. Do not modify.
VREF_EN	0	rw	0x0	Enables VREF internal generator

Register ([slcr](#)) GPIOB_CFG_CMOS18

Name	GPIOB_CFG_CMOS18
Relative Address	0x00000B04
Absolute Address	0xF8000B04
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	MIO GPIOB CMOS 1.8V config

Register GPIOB_CFG_CMOS18 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:28	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	27:25	rw	0x0	Reserved. Do not modify.
reserved	24:22	rw	0x0	Reserved. Do not modify.
reserved	21:19	rw	0x0	Reserved. Do not modify.
reserved	18:16	rw	0x0	Reserved. Do not modify.
reserved	15:12	rw	0x0	Reserved. Do not modify.
reserved	11:8	rw	0x0	Reserved. Do not modify.
reserved	7:4	rw	0x0	Reserved. Do not modify.
reserved	3:0	rw	0x0	Reserved. Do not modify.

Register ([slcr](#)) GPIOB_CFG_CMOS25

Name	GPIOB_CFG_CMOS25
Relative Address	0x00000B08
Absolute Address	0xF8000B08
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	MIO GPIOB CMOS 2.5V config

Register GPIOB_CFG_CMOS25 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:28	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	27:25	rw	0x0	Reserved. Do not modify.
reserved	24:22	rw	0x0	Reserved. Do not modify.
reserved	21:19	rw	0x0	Reserved. Do not modify.
reserved	18:16	rw	0x0	Reserved. Do not modify.
reserved	15:12	rw	0x0	Reserved. Do not modify.
reserved	11:8	rw	0x0	Reserved. Do not modify.
reserved	7:4	rw	0x0	Reserved. Do not modify.
reserved	3:0	rw	0x0	Reserved. Do not modify.

Register ([slcr](#)) GPIOB_CFG_CMOS33

Name	GPIOB_CFG_CMOS33
Relative Address	0x00000B0C
Absolute Address	0xF8000B0C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	MIO GPIOB CMOS 3.3V config

Register GPIOB_CFG_CMOS33 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:28	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	27:25	rw	0x0	Reserved. Do not modify.

Field Name	Bits	Type	Reset Value	Description
reserved	24:22	rw	0x0	Reserved. Do not modify.
reserved	21:19	rw	0x0	Reserved. Do not modify.
reserved	18:16	rw	0x0	Reserved. Do not modify.
reserved	15:12	rw	0x0	Reserved. Do not modify.
reserved	11:8	rw	0x0	Reserved. Do not modify.
reserved	7:4	rw	0x0	Reserved. Do not modify.
reserved	3:0	rw	0x0	Reserved. Do not modify.

Register ([slcr](#)) GPIOB_CFG_LVTTL

Name	GPIOB_CFG_LVTTL
Relative Address	0x00000B10
Absolute Address	0xF8000B10
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	MIO GPIOB LVTTL config

Register GPIOB_CFG_LVTTL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:28	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	27:25	rw	0x0	Reserved. Do not modify.
reserved	24:22	rw	0x0	Reserved. Do not modify.
reserved	21:19	rw	0x0	Reserved. Do not modify.
reserved	18:16	rw	0x0	Reserved. Do not modify.
reserved	15:12	rw	0x0	Reserved. Do not modify.
reserved	11:8	rw	0x0	Reserved. Do not modify.
reserved	7:4	rw	0x0	Reserved. Do not modify.
reserved	3:0	rw	0x0	Reserved. Do not modify.

Register ([slcr](#)) GPIOB_CFG_HSTL

Name	GPIOB_CFG_HSTL
Relative Address	0x00000B14
Absolute Address	0xF8000B14

Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	MIO GPIOB HSTL config

Register GPIOB_CFG_HSTL Details

You must provide a VREF or use the internal VREF generator.

When setting the input to HSTL, you must ensure that

VCCO_MIO is below 1.8V. If not, this will lead to long term damage to the IO.

Field Name	Bits	Type	Reset Value	Description
reserved	31:28	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	27:25	rw	0x0	Reserved. Do not modify.
reserved	24:22	rw	0x0	Reserved. Do not modify.
reserved	21:19	rw	0x0	Reserved. Do not modify.
reserved	18:16	rw	0x0	Reserved. Do not modify.
reserved	15:12	rw	0x0	Reserved. Do not modify.
reserved	11:8	rw	0x0	Reserved. Do not modify.
reserved	7:4	rw	0x0	Reserved. Do not modify.
reserved	3:0	rw	0x0	Reserved. Do not modify.

Register ([slcr](#)) GPIOB_DRVR_BIAS_CTRL

Name	GPIOB_DRVR_BIAS_CTRL
Relative Address	0x00000B18
Absolute Address	0xF8000B18
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	MIO GPIOB Driver Bias Control

Register GPIOB_DRVR_BIAS_CTRL Details

Field Name	Bits	Type	Reset Value	Description
RB_VCFG	31	ro	0x0	Right Bank VCFG (Read Only)
RB_DRVR_BIAS	30:16	rw	0x0	Right Bank driver bias control
LB_VCFG	15	ro	0x0	Left Bank VCFG (Read Only)
LB_DRVR_BIAS	14:0	rw	0x0	Left Bank driver bias control

Register ([slcr](#)) DDRIOB_ADDR0

Name	DDRIOB_ADDR0
Relative Address	0x00000B40
Absolute Address	0xF8000B40
Width	32 bits
Access Type	rw
Reset Value	0x00000800
Description	DDR IOB Config for Address 0

Register DDRIOB_ADDR0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:12	rw	0x0	Reserved
PULLUP_EN	11	rw	0x1	enables pullup on output 0: no pullup 1: pullup enabled
OUTPUT_EN	10:9	rw	0x0	Enables output mode to enable output ties to 00: ibuf 01 and 10: reserved 11: obuf
TERM_DISABLE_MODE	8	rw	0x0	Use dynamic_dci_ts to control dci 0: termination enabled 1: use 'dynamic_dci_ts' control termination
IBUF_DISABLE_MODE	7	rw	0x0	Use ibuf_disable_into control ibuf 0: ibuf is enabled 1: use ibuf_disable_in_to control enable Note that this must be "0" during DRAM init/training and can only be set to 1 after init/training completes.
DCI_TYPE	6:5	rw	0x0	DCI Update 00: DCI Disabled 01: DCI Drive (HSTL12_DCI) 10: reserved 11: DCI Termination (SSTL15_T_DCI)
TERM_EN	4	rw	0x0	Tri State Termination Enabled 0 - disabled 1 - enabled
DCI_UPDATE_B	3	rw	0x0	DCI Update Enabled 0 - disabled 1 - enabled

Field Name	Bits	Type	Reset Value	Description
INP_TYPE	2:1	rw	0x0	Input buffer controls. 00: Input off, reads 0. 01: Vref based differential receiver for SSTL, HSTL. 10: Differential input receiver. 11: LVCMOS receiver.
reserved	0	rw	0x0	Reserved. Do not modify.

Register ([slcr](#)) DDRIOB_ADDR1

Name	DDRIOB_ADDR1
Relative Address	0x00000B44
Absolute Address	0xF8000B44
Width	32 bits
Access Type	rw
Reset Value	0x00000800
Description	DDR IOB Config for Address 1

Register DDRIOB_ADDR1 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:12	rw	0x0	Reserved
PULLUP_EN	11	rw	0x1	enables pullup on output 0: no pullup 1: pullup enabled
OUTPUT_EN	10:9	rw	0x0	Enables output mode to enable output ties to 00: ibuf 01 and 10: reserved 11: obuf
TERM_DISABLE_MODE	8	rw	0x0	Use dynamic_dci_ts to control dci 0: termination enabled 1: use 'dynamic_dci_ts' control termination
IBUF_DISABLE_MODE	7	rw	0x0	Use ibuf_disable_into control ibuf 0: ibuf is enabled 1: use ibuf_disable_in_to control enable Note that this must be "0" during DRAM init/training and can only be set to 1 after init/training completes.

Field Name	Bits	Type	Reset Value	Description
DCI_TYPE	6:5	rw	0x0	DCI Update 00: DCI Disabled 01: DCI Drive (HSTL12_DCI) 10: reserved 11: DCI Termination (SSTL15_T_DCI)
TERM_EN	4	rw	0x0	Tri State Termination Enabled 0 - disabled 1 - enabled
DCI_UPDATE_B	3	rw	0x0	DCI Update Enabled 0 - disabled 1 - enabled
INP_TYPE	2:1	rw	0x0	Input buffer controls. 00: Input off, reads 0. 01: Vref based differential receiver for SSTL, HSTL. 10: Differential input receiver. 11: LVCMOS receiver.
reserved	0	rw	0x0	Reserved. Do not modify.

Register ([slcr](#)) DDRIOB_DATA0

Name	DDRIOB_DATA0
Relative Address	0x00000B48
Absolute Address	0xF8000B48
Width	32 bits
Access Type	rw
Reset Value	0x00000800
Description	DDR IOB Config for Data 15:0

Register DDRIOB_DATA0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:12	rw	0x0	Reserved
PULLUP_EN	11	rw	0x1	enables pullup on output 0: no pullup 1: pullup enabled
OUTPUT_EN	10:9	rw	0x0	Enables output mode to enable output ties to 00: ibuf 01 and 10: reserved 11: obuf

Field Name	Bits	Type	Reset Value	Description
TERM_DISABLE_MODE	8	rw	0x0	Use dynamic_dci_ts to control dci 0: termination enabled 1: use 'dynamic_dci_ts' control termination
IBUF_DISABLE_MODE	7	rw	0x0	Use ibuf_disable_into control ibuf 0: ibuf is enabled 1: use ibuf_disable_in_to control enable Note that this must be "0" during DRAM init/training and can only be set to 1 after init/training completes.
DCI_TYPE	6:5	rw	0x0	DCI Update 00: DCI Disabled 01: DCI Drive (HSTL12_DCI) 10: reserved 11: DCI Termination (SSTL15_T_DCI)
TERM_EN	4	rw	0x0	Tri State Termination Enabled 0 - disabled 1 - enabled
DCI_UPDATE_B	3	rw	0x0	DCI Update Enabled 0 - disabled 1 - enabled
INP_TYPE	2:1	rw	0x0	Input buffer controls. 00: Input off, reads 0. 01: Vref based differential receiver for SSTL, HSTL. 10: Differential input receiver. 11: LVCMOS receiver.
reserved	0	rw	0x0	Reserved. Do not modify.

Register ([slcr](#)) DDRIOB_DATA1

Name	DDRIOB_DATA1
Relative Address	0x00000B4C
Absolute Address	0xF8000B4C
Width	32 bits
Access Type	rw
Reset Value	0x00000800
Description	DDR IOB Config for Data 31:16

Register DDRIOB_DATA1 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:12	rw	0x0	Reserved
PULLUP_EN	11	rw	0x1	enables pullup on output 0: no pullup 1: pullup enabled
OUTPUT_EN	10:9	rw	0x0	Enables output mode to enable output ties to 00: ibuf 01 and 10: reserved 11: obuf
TERM_DISABLE_MODE	8	rw	0x0	Use dynamic_dci_ts to control dci 0: termination enabled 1: use 'dynamic_dci_ts' control termination
IBUF_DISABLE_MODE	7	rw	0x0	Use ibuf_disable_into control ibuf 0: ibuf is enabled 1: use ibuf_disable_in_to control enable Note that this must be "0" during DRAM init/training and can only be set to 1 after init/training completes.
DCI_TYPE	6:5	rw	0x0	DCI Update 00: DCI Disabled 01: DCI Drive (HSTL12_DCI) 10: reserved 11: DCI Termination (SSTL15_T_DCI)
TERM_EN	4	rw	0x0	Tri State Termination Enabled 0 - disabled 1 - enabled
DCI_UPDATE_B	3	rw	0x0	DCI Update Enabled 0 - disabled 1 - enabled
INP_TYPE	2:1	rw	0x0	Input buffer controls. 00: Input off, reads 0. 01: Vref based differential receiver for SSTL, HSTL. 10: Differential input receiver. 11: LVCMOS receiver.
reserved	0	rw	0x0	Reserved. Do not modify.

Register ([slcr](#)) DDRIOB_DIFF0

Name	DDRIOB_DIFF0
Relative Address	0x00000B50
Absolute Address	0xF8000B50

Width	32 bits
Access Type	rw
Reset Value	0x00000800
Description	DDR IOB Config for DQS 1:0

Register DDRIOB_DIFF0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:12	rw	0x0	Reserved
PULLUP_EN	11	rw	0x1	enables pullup on output 0: no pullup 1: pullup enabled
OUTPUT_EN	10:9	rw	0x0	Enables output mode to enable output ties to 00: ibuf 01 and 10: reserved 11: obuf
TERM_DISABLE_MODE	8	rw	0x0	Use dynamic_dci_ts to control dci 0: termination enabled 1: use 'dynamic_dci_ts' control termination
IBUF_DISABLE_MODE	7	rw	0x0	Use ibuf_disable_into control ibuf 0: ibuf is enabled 1: use ibuf_disable_in_to control enable Note that this must be "0" during DRAM init/training and can only be set to 1 after init/training completes.
DCI_TYPE	6:5	rw	0x0	DCI Update 00: DCI Disabled 01: DCI Drive (HSTL12_DCI) 10: reserved 11: DCI Termination (SSTL15_T_DCI)
TERM_EN	4	rw	0x0	Tri State Termination Enabled 0 - disabled 1 - enabled
DCI_UPDATE_B	3	rw	0x0	DCI Update Enabled 0 - disabled 1 - enabled
INP_TYPE	2:1	rw	0x0	Input buffer controls. 00: Input off, reads 0. 01: Vref based differential receiver for SSTL, HSTL. 10: Differential input receiver. 11: LVCMOS receiver.
reserved	0	rw	0x0	Reserved. Do not modify.

Register ([slcr](#)) DDRIOB_DIFF1

Name	DDRIOB_DIFF1
Relative Address	0x00000B54
Absolute Address	0xF8000B54
Width	32 bits
Access Type	rw
Reset Value	0x00000800
Description	DDR IOB Config for DQS 3:2

Register DDRIOB_DIFF1 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:12	rw	0x0	Reserved
PULLUP_EN	11	rw	0x1	enables pullup on output 0: no pullup 1: pullup enabled
OUTPUT_EN	10:9	rw	0x0	Enables output mode to enable output ties to 00: ibuf 01 and 10: reserved 11: obuf
TERM_DISABLE_MODE	8	rw	0x0	Use dynamic_dci_ts to control dci 0: termination enabled 1: use 'dynamic_dci_ts' control termination
IBUF_DISABLE_MODE	7	rw	0x0	Use ibuf_disable_into control ibuf 0: ibuf is enabled 1: use ibuf_disable_in_to control enable Note that this must be "0" during DRAM init/training and can only be set to 1 after init/training completes.
DCI_TYPE	6:5	rw	0x0	DCI Update 00: DCI Disabled 01: DCI Drive (HSTL12_DCI) 10: reserved 11: DCI Termination (SSTL15_T_DCI)
TERM_EN	4	rw	0x0	Tri State Termination Enabled 0 - disabled 1 - enabled
DCI_UPDATE_B	3	rw	0x0	DCI Update Enabled 0 - disabled 1 - enabled

Field Name	Bits	Type	Reset Value	Description
INP_TYPE	2:1	rw	0x0	Input buffer controls. 00: Input off, reads 0. 01: Vref based differential receiver for SSTL, HSTL. 10: Differential input receiver. 11: LVCMOS receiver.
reserved	0	rw	0x0	Reserved. Do not modify.

Register ([slcr](#)) DDRIOB_CLOCK

Name	DDRIOB_CLOCK
Relative Address	0x0000B58
Absolute Address	0xF800B58
Width	32 bits
Access Type	rw
Reset Value	0x00000800
Description	DDR IOB Config for Clock Output

Register DDRIOB_CLOCK Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:12	rw	0x0	Reserved
PULLUP_EN	11	rw	0x1	enables pullup on output 0: no pullup 1: pullup enabled
OUTPUT_EN	10:9	rw	0x0	Enables output mode to enable output ties to 00: ibuf 01 and 10: reserved 11: obuf
TERM_DISABLE_MODE	8	rw	0x0	Use dynamic_dci_ts to control dci 0: termination enabled 1: use 'dynamic_dci_ts' control termination
IBUF_DISABLE_MODE	7	rw	0x0	Use ibuf_disable_into control ibuf 0: ibuf is enabled 1: use ibuf_disable_in_to control enable Note that this must be "0" during DRAM init/training and can only be set to 1 after init/training completes.

Field Name	Bits	Type	Reset Value	Description
DCI_TYPE	6:5	rw	0x0	DCI Update 00: DCI Disabled 01: DCI Drive (HSTL12_DCI) 10: reserved 11: DCI Termination (SSTL15_T_DCI)
TERM_EN	4	rw	0x0	Tri State Termination Enabled 0 - disabled 1 - enabled
DCI_UPDATE_B	3	rw	0x0	DCI Update Enabled 0 - disabled 1 - enabled
INP_TYPE	2:1	rw	0x0	Input buffer controls. 00: Input off, reads 0. 01: Vref based differential receiver for SSTL, HSTL. 10: Differential input receiver. 11: LVCMOS receiver.
reserved	0	rw	0x0	Reserved. Do not modify.

Register ([slcr](#)) DDRIOB_DRIVE_SLEW_ADDR

Name	DDRIOB_DRIVE_SLEW_ADDR
Relative Address	0x00000B5C
Absolute Address	0xF8000B5C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	DDR IOB Slew for Address

Register DDRIOB_DRIVE_SLEW_ADDR Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:27	rw	0x0	Reserved. Do not modify.
reserved	26:24	rw	0x0	Reserved. Do not modify.
SLEW_N	23:19	rw	0x0	DDRIO slew rate for the N devices
SLEW_P	18:14	rw	0x0	DDRIO slew rate for the P devices
DRIVE_N	13:7	rw	0x0	DDRIO drive strength for the N devices
DRIVE_P	6:0	rw	0x0	DDRIO drive strength for the P devices

Register ([slcr](#)) DDRIOB_DRIVE_SLEW_DATA

Name	DDRIOB_DRIVE_SLEW_DATA
Relative Address	0x00000B60
Absolute Address	0xF8000B60
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	DDR IOB Slew for Data

Register DDRIOB_DRIVE_SLEW_DATA Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:27	rw	0x0	Reserved. Do not modify.
reserved	26:24	rw	0x0	Reserved. Do not modify.
SLEW_N	23:19	rw	0x0	DDRIO slew rate for the N devices
SLEW_P	18:14	rw	0x0	DDRIO slew rate for the P devices
DRIVE_N	13:7	rw	0x0	DDRIO drive strength for the N devices
DRIVE_P	6:0	rw	0x0	DDRIO drive strength for the P devices

Register ([slcr](#)) DDRIOB_DRIVE_SLEW_DIFF

Name	DDRIOB_DRIVE_SLEW_DIFF
Relative Address	0x00000B64
Absolute Address	0xF8000B64
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	DDR IOB Slew for Diff

Register DDRIOB_DRIVE_SLEW_DIFF Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:27	rw	0x0	Reserved. Do not modify.
reserved	26:24	rw	0x0	Reserved. Do not modify.
SLEW_N	23:19	rw	0x0	DDRIO slew rate for the N devices
SLEW_P	18:14	rw	0x0	DDRIO slew rate for the P devices

Field Name	Bits	Type	Reset Value	Description
DRIVE_N	13:7	rw	0x0	DDRIO drive strength for the N devices
DRIVE_P	6:0	rw	0x0	DDRIO drive strength for the P devices

Register ([slcr](#)) DDRIOB_DRIVE_SLEW_CLOCK

Name	DDRIOB_DRIVE_SLEW_CLOCK
Relative Address	0x00000B68
Absolute Address	0xF8000B68
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	DDR IOB Slew for Clock

Register DDRIOB_DRIVE_SLEW_CLOCK Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:27	rw	0x0	Reserved. Do not modify.
reserved	26:24	rw	0x0	Reserved. Do not modify.
SLEW_N	23:19	rw	0x0	DDRIO slew rate for the N devices
SLEW_P	18:14	rw	0x0	DDRIO slew rate for the P devices
DRIVE_N	13:7	rw	0x0	DDRIO drive strength for the N devices
DRIVE_P	6:0	rw	0x0	DDRIO drive strength for the P devices

Register ([slcr](#)) DDRIOB_DDR_CTRL

Name	DDRIOB_DDR_CTRL
Relative Address	0x00000B6C
Absolute Address	0xF8000B6C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	DDR IOB Buffer Control

Register DDRIOB_DDR_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:15	rw	0x0	reserved
reserved	14	rw	0x0	Reserved. Do not modify.
reserved	13	rw	0x0	Reserved. Do not modify.
reserved	12	rw	0x0	Reserved. Do not modify.
reserved	11:10	rw	0x0	Reserved. Do not modify.
REFIO_EN	9	rw	0x0	Enables VRP,VRN 0: VRP/VRN not used 1: VRP/VRN used as refio
reserved	8:7	rw	0x0	Reserved. Do not modify.
VREF_EXT_EN	6:5	rw	0x0	Enables External VREF input x0: Disable External VREF for lower 16 bits x1: Enable External VREF for lower 16 bits 0x: Disable External VREF for upper 16 bits 1X: Enable External VREF for upper 16 bits
VREF_SEL	4:1	rw	0x0	Specifies DDR IOB Vref generator output: 0001: VREF = 0.6V for LPDDR2 with 1.2V IO 0100: VREF = 0.75V for DDR3 with 1.5V IO 1000: VREF = 0.90V for DDR2 with 1.8V IO
VREF_INT_EN	0	rw	0x0	Enables VREF internal generator

Register ([slcr](#)) DDRIOB_DCI_CTRL

Name	DDRIOB_DCI_CTRL
Relative Address	0x00000B70
Absolute Address	0xF8000B70
Width	32 bits
Access Type	rw
Reset Value	0x00000020
Description	DDR IOB DCI Config

Register DDRIOB_DCI_CTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:28	rw	0x0	Reserved. Writes are ignored, read data is zero.
reserved	27	rw	0x0	Reserved. Do not modify.
reserved	26	rw	0x0	Reserved. Do not modify.

Field Name	Bits	Type	Reset Value	Description
reserved	25	rw	0x0	Reserved. Do not modify.
reserved	24	rw	0x0	Reserved. Do not modify.
reserved	23	rw	0x0	Reserved. Do not modify.
reserved	22	rw	0x0	Reserved. Do not modify.
reserved	21	rw	0x0	Reserved. Do not modify.
UPDATE_CONTROL	20	rw	0x0	DCI Update
PREF_OPT2	19:17	rw	0x0	DCI Calibration. See Table 10-7 of Zynq TRM (UG585) for mode specific values
reserved	16	rw	0x0	Reserved
PREF_OPT1	15:14	rw	0x0	DCI Calibration. See Table 10-7 of Zynq TRM (UG585) for mode specific values
NREF_OPT4	13:11	rw	0x0	DCI Calibration. See Table 10-7 of Zynq TRM (UG585) for mode specific values
NREF_OPT2	10:8	rw	0x0	DCI Calibration. See Table 10-7 of Zynq TRM (UG585) for mode specific values
NREF_OPT1	7:6	rw	0x0	DCI Calibration. See Table 10-7 of Zynq TRM (UG585) for mode specific values
reserved	5	rw	0x1	Reserved. Do not modify.
reserved	4	rw	0x0	Reserved. Do not modify.
reserved	3	rw	0x0	Reserved. Do not modify.
reserved	2	rw	0x0	Reserved. Do not modify.
ENABLE	1	rw	0x0	1 if any iob's use a terminate type, or if dci test block used
RESET	0	rw	0x0	At least toggle once to initialise flops in DCI system

Register ([slcr](#)) DDRIOB_DCI_STATUS

Name	DDRIOB_DCI_STATUS
Relative Address	0x00000B74
Absolute Address	0xF8000B74
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	DDR IO Buffer DCI Status

Register DDRIOB_DCI_STATUS Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:14	ro	0x0	Reserved. Writes are ignored, read data is zero.
DONE	13	rw	0x0	DCI done signal
reserved	12	rw	0x0	Reserved. Do not modify.
reserved	11	rw	0x0	Reserved. Do not modify.
reserved	10	ro	0x0	Reserved. Do not modify.
reserved	9	ro	0x0	Reserved. Do not modify.
reserved	8	ro	0x0	Reserved. Do not modify.
reserved	7	ro	0x0	Reserved. Do not modify.
reserved	6	ro	0x0	Reserved. Do not modify.
reserved	5	ro	0x0	Reserved. Do not modify.
reserved	4:3	ro	0x0	Reserved. Do not modify.
reserved	2	ro	0x0	Reserved. Do not modify.
reserved	1	ro	0x0	Reserved. Do not modify.
LOCK	0	ro	0x0	DCI Status input Read Only

B.29 Static Memory Controller (pl353)

Module Name	Static Memory Controller (pl353)
Software Name	XNANDPS
Base Address	0xE000E000 smcc
Description	Shared memory controller
Version	1.0
Doc Version	1.0
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
memc_status	0x00000000	13	ro	0x00000000	Operating and Interrupt Status
memif_cfg	0x00000004	18	ro	0x00011205	SMC configuration information, read-only
memc_cfg_set	0x00000008	7	wo	x	Enable interrupts and lower power state
memc_cfg_clr	0x0000000C	7	wo	x	Disable interrupts and exit from low-power state
direct_cmd	0x00000010	26	wo	x	Issue mem commands and register updates
set_cycles	0x00000014	24	wo	x	Stage a write to a Cycle register
set_opmode	0x00000018	16	mixed	x	Stage a write to an OpMode register
refresh_period_0	0x00000020	4	rw	0x00000000	Idle cycles between read/write bursts
refresh_period_1	0x00000024	4	rw	0x00000000	Insert idle cycles between bursts
sram_cycles0_0	0x00000100	21	ro	0x0002B3CC	SRAM/NOR chip select 0 timing, active
opmode0_0	0x00000104	32	ro	0xE2FE0800	SRAM/NOR chip select 0 OpCode, active
sram_cycles0_1	0x00000120	21	ro	0x0002B3CC	SRAM/NOR chip select 1 timing, active
opmode0_1	0x00000124	32	ro	0xE4FE0800	SRAM/NOR chip select 1 OpCode, active
nand_cycles1_0	0x00000180	24	ro	0x0024ABCC	NAND Flash timing, active

Register Name	Address	Width	Type	Reset Value	Description
opmode1_0	0x00000184	32	ro	0xE1FF0001	NAND Flash OpCode, active
user_status	0x00000200	8	ro	0x00000000	The user_status is read-only and returns the value of the user_status[7:0] signals. You can read this register in all operating states.
user_config	0x00000204	8	wo	x	The user_config is write-only and controls the status of the user_config[7:0] signals. You can write to this register in all operating states.
ecc_status_1	0x00000400	30	ro	0x00000000	ECC status and clear
ecc_memcfg_1	0x00000404	13	rw	0x00000043	ECC Memory Config
ecc_memcommand1_1	0x00000408	25	rw	0x01300080	Commands used to detect start of ECC operation
ecc_memcommand2_1	0x0000040C	25	rw	0x01E00585	Commands used to access memory within a page
ecc_addr0_1	0x00000410	32	ro	0x00000000	Lower 32 bits of ECC address
ecc_addr1_1	0x00000414	24	ro	0x00000000	Upper 24 bits of ECC address
ecc_value0_1	0x00000418	32	ro	0x00000000	The five ecc_value Registers are read-only and contain block information for the ECC. Note: Writing any value to an ecc_value Register clears the ecc_int bit.
ecc_value1_1	0x0000041C	32	ro	0x00000000	The five ecc_value Registers are read-only and contain block information for the ECC. Note: writing any value to an ecc_value Register clears the ecc_int bit.
ecc_value2_1	0x00000420	32	ro	0x00000000	The five ecc_value Registers are read-only and contain block information for the ECC. Note: writing any value to an ecc_value Register clears the ecc_int bit.
ecc_value3_1	0x00000424	32	ro	0x00000000	The five ecc_value Registers are read-only and contain block information for the ECC. Note: writing any value to an ecc_value Register clears the ecc_int bit.

Register ([p1353](#)) memc_status

Name	memc_status
Software Name	MEMC_STATUS
Relative Address	0x00000000
Absolute Address	0xE000E000
Width	13 bits
Access Type	ro
Reset Value	0x00000000
Description	Operating and Interrupt Status

Register memc_status Details

The read-only memc_status Register provides information on the configuration of the SMC and also the current state of the SMC. You cannot read this register in the Reset state

Field Name	Bits	Type	Reset Value	Description
raw_ecc_int1 (RAW_ECC_INT1)	12	ro	0x0	NAND Flash ECC interrupt raw status: 0: not asserted, 1: asserted
reserved	11	ro	0x0	Reserved. Do not modify.
ecc_int1 (ECC_INT1)	10	ro	0x0	NAND Flash ECC interrupt status after mask/enable: 0: not asserted, 1: asserted
reserved	9	ro	0x0	Reserved. Do not modify.
ecc_int1_en (ECC_INT_EN1)	8	ro	0x0	NAND Flash ECC interrupt enable setting: 0: masked, 1: enabled
reserved	7	ro	0x0	Reserved. Do not modify.
raw_int_status1 (RAW_INT_STATUS1)	6	ro	0x0	NAND Flash raw interrupt status before mask/enable: 0: not asserted, 1: asserted
raw_int_status0 (RAW_INT_STATUS0)	5	ro	0x0	SRAM/NOR raw interrupt raw status before the mask/enable: 0: not asserted, 1: asserted
int_status1 (INT_STATUS1)	4	ro	0x0	NAND Flash interrupt status after the mask/enable: 0: not asserted, 1: asserted
int_status0 (INT_STATUS0)	3	ro	0x0	SRAM/NOR interrupt status after the mask/enable : 0: not asserted, 1: asserted
int_en1 (INT_EN1)	2	ro	0x0	NAND Flash interrupt enable status: 0: disabled, 1: enabled

Field Name	Bits	Type	Reset Value	Description
int_en0 (INT_EN0)	1	ro	0x0	SRAM/NOR interface interrupt enable setting: 0: disabled, 1: enabled
state (STATE)	0	ro	0x0	SMC operating state: 0: normal, 1: low-power state

Register (p353) memif_cfg

Name	memif_cfg
Software Name	MEMC_IF_CONFIG
Relative Address	0x00000004
Absolute Address	0xE000E004
Width	18 bits
Access Type	ro
Reset Value	0x00011205
Description	SMC configuration information, read-only

Register memif_cfg Details

Provides information on the configuration of the memory interface. You cannot read this register in the Reset state. The state of this register cannot be changed.

Field Name	Bits	Type	Reset Value	Description
exclusive_monitors (EX_MONITORS)	17:16	ro	0x1	Return the number of exclusive access monitor resources that are implemented in the SMC. B00: 0 monitors b01: 1 monitor b10: 2 monitors b11: 4 monitors
reserved	15	ro	0x0	Reserved
remap1 (REMAP1)	14	ro	0x0	Return the value of the remap_1 input.
memory_width1 (MEMORY_WIDTH1)	13:12	ro	0x1	The width of the NAND Flash interface can be 8 or 16 bits.
memory_chips1 (MEMORY_CHIPS1)	11:10	ro	0x0	The NAND Flash interface provides one chip select.
memory_type1 (MEMORY_TYPE1)	9:8	ro	0x2	SMC controller 1 supports the NAND Flash interface with hardware assisted ECC.
reserved	7	ro	0x0	Reserved

Field Name	Bits	Type	Reset Value	Description
remap0 (REMAP0)	6	ro	0x0	Return the value of the remap_0 input: 0: 1:
memory_width0 (MEMORY_WIDTH0)	5:4	ro	0x0	The width of the SRAM/NOR interface is 8 bits.
memory_chips0 (MEMORY_CHIPS0)	3:2	ro	0x1	The SRAM/NOR interface provides two chip selects.
memory_type0 (MEMORY_TYPE0)	1:0	ro	0x1	SMC controller 0 supports the SRAM/NOR interface.

Register (p1353) memc_cfg_set

Name	memc_cfg_set
Software Name	MEMC_SET_CONFIG
Relative Address	0x00000008
Absolute Address	0xE000E008
Width	7 bits
Access Type	wo
Reset Value	x
Description	Enable interrupts and lower power state

Register memc_cfg_set Details

The write-only memc_cfg_set enables the SMC to be changed to low-power state, and interrupts enabled. You cannot write to this register in the Reset state.

Field Name	Bits	Type	Reset Value	Description
ecc_int_enable1 (ECC_INT_ENABLE1)	6	wo	x	NAND Flash ECC interrupt enable: 0: No change, 1: enable
reserved	5	wo	x	Reserved. Do not modify.
reserved	4:3	wo	x	Reserved, write as zero.
low_power_req (LOW_POWER_REQ)	2	wo	x	Put SMC into low-power mode when memory interface goes idle: 0: No change, 1: enable low-power state
int_enable1 (INT_ENABLE1)	1	wo	x	NAND Flash interrupt enable: 0: No change, 1: enable
int_enable0 (INT_ENABLE0)	0	wo	x	SRAM/NOR interrupt enable: 0: No change, 1: enable

Register ([p1353](#)) memc_cfg_clr

Name	memc_cfg_clr
Software Name	MEMC_CLR_CONFIG
Relative Address	0x0000000C
Absolute Address	0xE000E00C
Width	7 bits
Access Type	wo
Reset Value	x
Description	Disable interrupts and exit from low-power state

Register memc_cfg_clr Details

The write-only memc_cfg_clr enables the SMC to be moved out of the low-power state, and the interrupts disabled. You cannot write to this register in the Reset state.

Field Name	Bits	Type	Reset Value	Description
ecc_int_disable1 (ECC_INT_DISABLE1)	6	wo	x	NAND Flash ECC interrupt disable: 0: No change, 1: disable
reserved	5	wo	x	Reserved. Do not modify.
int_clr_1 (INT_CLR1)	4	wo	x	0: No effect 1: Clear SMC Interrupt 1 as an alternative to an AXI read
int_clr_0 (INT_CLR0)	3	wo	x	0: No effect 1: Clear SMC Interrupt 0 as an alternative to an AXI read
low_power_exit (LOW_POWER_EXIT)	2	wo	x	Exit low-power mode. The affect takes place when memory interface goes idle: 0: No change, 1: exit from low-power state
int_disable1 (INT_DISABLE1)	1	wo	x	NAND Flash interrupt disable: 0: No change, 1: disable (apply mask)
int_disable0 (INT_DISABLE0)	0	wo	x	SRAM/NOR interrupt disable: 0: No change, 1: disable (apply mask)

Register ([p1353](#)) direct_cmd

Name	direct_cmd
Software Name	DIRECT_CMD
Relative Address	0x00000010
Absolute Address	0xE000E010
Width	26 bits

Access Type	wo
Reset Value	x
Description	Issue mem commands and register updates

Register direct_cmd Details

The write-only direct_cmd passes commands to the external memory, and controls the updating of the chip configuration registers with values held in the set_cycles Register and set_opmode Register. You cannot write to this register in either the Reset or low-power states.

Field Name	Bits	Type	Reset Value	Description
chip_select (CHIP_SELECT)	25:23	wo	x	Select register bank to update and enable chip mode register access based on CMD_TYPE: 000: SRAM/NOR chip select 0. 001: SRAM/NOR chip select 1. 100: NAND Flash. others: reserved.
cmd_type (TYPE)	22:21	wo	x	Select the command type: 00: UpdateRegs and AXI 01: ModeReg 10: UpdateRegs 11: ModeReg and UpdateRegs
reserved	20	wo	x	Reserved. Do not modify.
addr (ADDR)	19:0	wo	x	When cmd_type = UpdateRegs and AXI then: Bits [15:0] are used to match wdata[15:0] Bits [19:16] are reserved. Write as zero. When cmd_type = ModeReg or ModeReg and UpdateRegs, these bits map to the external memory address bits [19:0]. When cmd_type = UpdateRegs, these bits are reserved. Write as zero.

Register ([p1353](#)) set_cycles

Name	set_cycles
Software Name	SET_CYCLES
Relative Address	0x00000014
Absolute Address	0xE000E014
Width	24 bits
Access Type	wo
Reset Value	x
Description	Stage a write to a Cycle register

Register set_cycles Details

This write-only register contains values that are written to the `sram_cycles` register or `nand_cycles` when the SMC receives a write to the Direct Command Register. You cannot write to this register in either the Reset or low-power states.

Field Name	Bits	Type	Reset Value	Description
Set_t6 (SET_T6)	23:20	wo	x	Timing parameter for SRAM/NOR, bit 20 only (other bits are ignored): o For asynchronous multiplexed transfers this bit controls when the SMC asserts <code>we_n</code> : 0: assert <code>we_n</code> two <code>mclk</code> cycles after asserting <code>cs_n</code> . 1: assert <code>we_n</code> and <code>cs_n</code> together. Timing parameter for NAND Flash, bits 23:20: o Busy to RE timing (<code>t_rr</code>), minimum permitted value = 0.
Set_t5 (SET_T5)	19:17	wo	x	Timing parameter for SRAM/NOR: o Turnaround time (<code>t_ta</code>), minimum value = 1. Timing parameter for NAND Flash: o ID read time (<code>t_ar</code>), minimum value = 0.
Set_t4 (SET_T4)	16:14	wo	x	Timing parameter for SRAM/NOR: o Page cycle time (<code>t_pc</code>), minimum value = 1. Timing parameter for NAND Flash: o Page cycle time (<code>t_clr</code>), minimum value = 1.
Set_t3 (SET_T3)	13:11	wo	x	Timing parameter for SRAM/NOR: o Write Enable (<code>t_wp</code>) assertion delay, minimum value = 1. Timing parameter for NAND Flash: o Write Enable (<code>t_wp</code>) deassertion delay, minimum value = 1.
Set_t2 (SET_T2)	10:8	wo	x	Timing parameter for SRAM/NOR: o Output Enable (<code>t_ceoe</code>) assertion delay, minimum value = 1. Timing parameter for NAND Flash: o REA (<code>t_rea</code>) assertion delay, minimum value = 1.
Set_t1 (SET_T1)	7:4	wo	x	Timing parameter for SRAM/NOR and NAND Flash: Write cycle time, minimum value = 2.
Set_t0 (SET_T0)	3:0	wo	x	Timing parameter for SRAM/NOR and NAND Flash: Read cycle time, minimum value = 2.

Register ([p353](#)) set_opmode

Name	set_opmode
Software Name	SET_OPMODE
Relative Address	0x00000018
Absolute Address	0xE000E018
Width	16 bits
Access Type	mixed
Reset Value	x
Description	Stage a write to an OpMode register

Register set_opmode Details

This write-only register is the holding register for the opmode<x>_<n> Registers. You cannot write to it in either the Reset or low-power states.

Field Name	Bits	Type	Reset Value	Description
set_burst_align (SET_BURST_ALIGN)	15:13	wo	x	NAND Flash: reserved, write zero. SRAM/NOR: Value written to the burst_align field. When SMC is configured to perform synchronous transfers, these bits control if memory bursts are split on memory burst boundaries: 000: bursts can cross any address boundary 001: burst split on memory burst boundary; that is, 32 beats for continuous 010: burst split on 64 beat boundary 011: burst split on 128 beat boundary 100: burst split on 256 beat boundary others: reserved
set_bls (SET_BLS)	12	wo	x	NAND Flash: reserved, write zero. SRAM/NOR: Value written to the byte lane strobe (bls) bit. This bit affects the assertion of the byte-lane strobe outputs. 0: bls timing equals chip select timing. This is the default setting. 1: bls timing equals we_n timing. This setting is used for eight memories that have no bls_n inputs. In this case, the bls_n output of the SMC is connected to the we_n memory input.
set_adv (SET_ADV)	11	wo	x	Contains the value to be written to the specific SRAM chip opmode Register address valid (adv) bit. The memory uses the address advance signal adv_n when set. For a NAND memory interface this bit is reserved, and written as zero.

Field Name	Bits	Type	Reset Value	Description
set_baa (SET_BAA)	10	rw	x	NAND Flash: reserved, write zero. SRAM/NOR: Value written burst address advance (baa) bit. The memory uses the baa_n signal when set.
set_wr_bl (SET_WR_BL)	9:7	wo	x	NAND Flash: reserved, write zero. SRAM/NORE: Value written for wr_bl : 000: 1 beat 001: 4 beats 010: 8 beats 011: 16 beats 100: 32 beats 101: continuous others: reserved.
set_wr_sync (SET_WR_SYNC)	6	wo	x	NAND Flash: reserved, write zero. SRAM/NOR: Write sync (wr_sync):. The memory writes are synchronous when set.
set_rd_bl (SET_RD_BL)	5:3	wo	x	NAND Flash: reserved, write zero. SRAM/NOR: value written to opmode (rd_bl field). Memory Burst Length: 000: 1 beat 001: 4 beats 010: 8 beats 011: 16 beats 100: 32 beats 101: continuous others: reserved
reserved	2	wo	x	Reserved. Do not modify.
set_mw (SET_MW)	1:0	wo	x	SRAM/NOR: mw= 00 (8-bit) NAND Flash: mw= 00 (8-bit) or 01 (16-bit)

Register ([pl353](#)) refresh_period_0

Name	refresh_period_0
Software Name	REFRESH_PERIOD_0
Relative Address	0x00000020
Absolute Address	0xE000E020
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	Idle cycles between read/write bursts

Register refresh_period_0 Details

The read/write refresh_period_0 enables the SMC to insert idle cycles during consecutive bursts, that enables the PSRAM devices on memory interface 0, to initiate a refresh cycle. You cannot access this register in either the Reset or low-power states. Note:

You can only access this register when you are using an SRAM memory interface.

Field Name	Bits	Type	Reset Value	Description
period (MASK)	3:0	rw	0x0	Set the number of consecutive memory bursts that are permitted, prior to the SMC deasserting chip select to enable the PSRAM to initiate a refresh cycle. The options are: b0000: disable the insertion of idle cycles between consecutive bursts b0001: an idle cycle occurs after each burst b0010: an idle cycle occurs after 2 consecutive bursts b0011: an idle cycle occurs after 3 consecutive bursts b0100: an idle cycle occurs after 4 consecutive bursts ... b1111: an idle cycle occurs after 15 consecutive bursts.

Register ([p353](#)) refresh_period_1

Name	refresh_period_1
Software Name	REFRESH_PERIOD_1
Relative Address	0x00000024
Absolute Address	0xE000E024
Width	4 bits
Access Type	rw
Reset Value	0x00000000
Description	Insert idle cycles between bursts

Register refresh_period_1 Details

The read/write refresh_period_1 Register enables the SMC to insert idle cycles during consecutive bursts, that enables the PSRAM devices on memory interface 1, to initiate a refresh cycle

Field Name	Bits	Type	Reset Value	Description
period (REFRESH_PERIOD_0)	3:0	rw	0x0	Set the number of consecutive memory bursts that are permitted, prior to the SMC deasserting chip select to enable the PSRAM to initiate a refresh cycle. The options are: b0000: disable the insertion of idle cycles between consecutive bursts b0001: an idle cycle occurs after each burst b0010: an idle cycle occurs after 2 consecutive bursts b0011: an idle cycle occurs after 3 consecutive bursts b0100: an idle cycle occurs after 4 consecutive bursts ... b1111: an idle cycle occurs after 15 consecutive bursts.

Register ([p1353](#)) sram_cycles0_0

Name	sram_cycles0_0
Software Name	IF0_CHIP_0_CONFIG
Relative Address	0x00000100
Absolute Address	0xE000E100
Width	21 bits
Access Type	ro
Reset Value	0x0002B3CC
Description	SRAM/NOR chip select 0 timing, active

Register sram_cycles0_0 Details

There is an instance of this register for each SRAM chip supported. You cannot read the read-only sram_cycles Register in the Reset state

Field Name	Bits	Type	Reset Value	Description
we_time	20	ro	0x0	Asynchronous assertion, refer to SET_CYCLES register.
t_tr	19:17	ro	0x1	Turnaround time, refer to SET_CYCLES register.
t_pc	16:14	ro	0x2	Page cycle time, refer to SET_CYCLES register.
t_wp	13:11	ro	0x6	WE assertion delay, refer to SET_CYCLES register.
t_ceoe	10:8	ro	0x3	OE assertion delay, refer to SET_CYCLES register.

Field Name	Bits	Type	Reset Value	Description
t_wc	7:4	ro	0xC	Write cycle time, refer to SET_CYCLES register.
t_rc	3:0	ro	0xC	Read cycle time, refer to SET_CYCLES register.

Register (p1353) opmode0_0

Name	opmode0_0
Software Name	OPMODE
Relative Address	0x00000104
Absolute Address	0xE000E104
Width	32 bits
Access Type	ro
Reset Value	0xE2FE0800
Description	SRAM/NOR chip select 0 OpCode, active

Register opmode0_0 Details

Field Name	Bits	Type	Reset Value	Description
address_match (ADDRESS_MATCH)	31:24	ro	0xE2	Return the value of this tie-off. This is the comparison value for address bits [31:24] to determine the chip that is selected.
address_mask (ADDRESS)	23:16	ro	0xFE	Return the value of this tie-off. This is the mask for address bits[31:24] to determine the chip that must be selected. A logic 1 indicates the bit is used for comparison. 0: 1:
reserved	15:13	ro	0x0	Reserved. Do not modify.
reserved	12	ro	0x0	Reserved. Do not modify.
reserved	11	ro	0x1	Reserved. Do not modify.
baa (BAA)	10	ro	0x0	The memory uses the burst address advance signal, baa_n, when set. For a NAND memory interface, this bit is reserved.
wr_bl (WR_BL)	9:7	ro	0x0	Selects the write burst lengths, see SET_OPMODE register.
reserved	6	ro	0x0	Reserved. Do not modify.
rd_bl (RD_BL)	5:3	ro	0x0	Select memory burst lengths, see SET_OPMODE Register.

Field Name	Bits	Type	Reset Value	Description
reserved	2	ro	0x0	Reserved. Do not modify.
mw (MW)	1:0	ro	0x0	Select data bus width (8 or 16), see SET_OPMODE register.

Register ([p1353](#)) sram_cycles0_1

Name	sram_cycles0_1
Software Name	IF0_CHIP_1_CONFIG
Relative Address	0x00000120
Absolute Address	0xE000E120
Width	21 bits
Access Type	ro
Reset Value	0x0002B3CC
Description	SRAM/NOR chip select 1 timing, active

Register sram_cycles0_1 Details

There is an instance of this register for each SRAM chip supported. You cannot read the read-only sram_cycles Register in the Reset state

Field Name	Bits	Type	Reset Value	Description
we_time	20	ro	0x0	Asynchronous assertion, refer to SET_CYCLES register.
t_tr	19:17	ro	0x1	Turnaround time, refer to SET_CYCLES register.
t_pc	16:14	ro	0x2	Page cycle time, refer to SET_CYCLES register.
t_wp	13:11	ro	0x6	WE assertion delay, refer to SET_CYCLES register.
t_ceoe	10:8	ro	0x3	OE assertion delay, refer to SET_CYCLES register.
t_wc	7:4	ro	0xC	Write cycle time, refer to SET_CYCLES register.
t_rc	3:0	ro	0xC	Read cycle time, refer to SET_CYCLES register.

Register ([p1353](#)) opmode0_1

Name	opmode0_1
Relative Address	0x00000124
Absolute Address	0xE000E124
Width	32 bits
Access Type	ro
Reset Value	0xE4FE0800

Description SRAM/NOR chip select 1 OpCode, active

Register opmode0_1 Details

Field Name	Bits	Type	Reset Value	Description
address_match (OPMODE_ADDRESS_MATCH)	31:24	ro	0xE4	see 0x120
address_mask (OPMODE_ADDRESS)	23:16	ro	0xFE	see 0x120
burst_align (OPMODE_BURST_ALIGN)	15:13	ro	0x0	reserved
bls (OPMODE_BLS)	12	ro	0x0	reserved
adv (OPMODE_ADV)	11	ro	0x1	reserved
baa (OPMODE_BAA)	10	ro	0x0	The memory uses the burst address advance signal, baa_n, when set. For a NAND memory interface, this bit is reserved.
wr_bl (OPMODE_WR_BL)	9:7	ro	0x0	Selects the write burst lengths, see SET_OPMODE register.
wr_sync (OPMODE_WR_SYNC)	6	ro	0x0	SRAM/NOR interface operates in asynchronous mode
rd_bl (OPMODE_RD_BL)	5:3	ro	0x0	Select memory burst lengths, see SET_OPMODE Register.
rd_sync (OPMODE_RD_SYNC)	2	ro	0x0	reserved
mw (OPMODE_MW)	1:0	ro	0x0	Data bus width (8 or 16), see SET_OPMODE register.

Register ([p1353](#)) nand_cycles1_0

Name nand_cycles1_0
Software Name IF1_CHIP_0_CONFIG
Relative Address 0x00000180
Absolute Address 0xE000E180
Width 24 bits
Access Type ro

Reset Value 0x0024ABCC

Description NAND Flash timing, active

Register nand_cycles1_0 Details

There is an instance of this register for each NAND chip supported. You cannot read the read-only nand_cycles Register in the Reset state

Field Name	Bits	Type	Reset Value	Description
t_rr	23:20	ro	0x2	BUSY to RE, refer to SET_CYCLES register.
t_ar	19:17	ro	0x2	ID read time, refer to SET_CYCLES register.
t_clr	16:14	ro	0x2	Page cycle time, refer to SET_CYCLES register. [RE: decipher: Status read time for NAND chip configurations. Minimum permitted value = 0.]
t_wp	13:11	ro	0x5	WE deassertion delay, refer to SET_CYCLES register.
t_rea	10:8	ro	0x3	RE assertion delay, refer to SET_CYCLES register.
t_wc	7:4	ro	0xC	Write cycle time, refer to SET_CYCLES register.
t_rc	3:0	ro	0xC	Read cycle time, refer to SET_CYCLES register.

Register (pl353) opmode1_0

Name opmode1_0

Relative Address 0x00000184

Absolute Address 0xE000E184

Width 32 bits

Access Type ro

Reset Value 0xE1FF0001

Description NAND Flash OpCode, active

Register opmode1_0 Details

Field Name	Bits	Type	Reset Value	Description
address_match (OPMODE_ADDRESS_MATCH)	31:24	ro	0xE1	Return the value of this tie-off. This is the comparison value for address bits [31:24] to determine the chip that is selected.
address_mask (OPMODE_ADDRESS)	23:16	ro	0xFF	Return the value of this tie-off. This is the mask for address bits[31:24] to determine the chip that must be selected. A logic 1 indicates the bit is used for comparison.
reserved	15:13	ro	0x0	Reserved. Do not modify.
reserved	12	ro	0x0	Reserved. Do not modify.

Field Name	Bits	Type	Reset Value	Description
reserved	11	ro	0x0	Reserved. Do not modify.
reserved	10	ro	0x0	Reserved. Do not modify.
reserved	9:7	ro	0x0	Reserved. Do not modify.
reserved	6	ro	0x0	Reserved. Do not modify.
reserved	5:3	ro	0x0	Reserved. Do not modify.
reserved	2	ro	0x0	Reserved. Do not modify.
mw (OPMODE_MW)	1:0	ro	0x1	Data bus width is 8 bits, see SET_OPMODE register.

Register (p1353) user_status

Name	user_status
Software Name	USER_STATUS
Relative Address	0x00000200
Absolute Address	0xE000E200
Width	8 bits
Access Type	ro
Reset Value	0x00000000
Description	The user_status is read-only and returns the value of the user_status[7:0] signals. You can read this register in all operating states.

Register user_status Details

Field Name	Bits	Type	Reset Value	Description
user_status (MASK)	7:0	ro	0x0	This value returns the state of the user_status[7:0] inputs.

Register (p1353) user_config

Name	user_config
Software Name	USER_CONFIG
Relative Address	0x00000204
Absolute Address	0xE000E204
Width	8 bits
Access Type	wo
Reset Value	x

Description The user_config is write-only and controls the status of the user_config[7:0] signals. You can write to this register in all operating states.

Register user_config Details

Field Name	Bits	Type	Reset Value	Description
user_config (MASK)	7:0	wo	x	This value sets the state of the user_config[7:0] outputs.

Register ([p1353](#)) ecc_status_1

Name ecc_status_1

Software Name IF1_ECC

Relative Address 0x00000400

Absolute Address 0xE000E400

Width 30 bits

Access Type ro

Reset Value 0x00000000

Description ECC status and clear

Register ecc_status_1 Details

The ecc_status

is read-only and contains status information for the ECC. Although this is a read-only register, the bottom five bits can be written to clear the corresponding interrupts. To clear the interrupt, you must write a 1 to the appropriate bit.

Field Name	Bits	Type	Reset Value	Description
ecc_read (ECC_READ)	29:25	ro	0x0	Read flags for ECC blocks. Indicate whether the stored ECC value for each block has been read from memory: 0: not read 1: read Bit [29] Extra block (if used). Bit [28] Block 3. Bit [27] Block 2. Bit [26] Block 1. Bit [25] Block 0.
ecc_can_correct (ECC_CAN_CORRECT)	24:20	ro	0x0	Correctable flag for each ECC block: 0: not correctable error 1: correctable error Bit [24] Extra block (if used). Bit [23] Block 3. Bit [22] Block 2. Bit [21] Block 1. Bit [20] Block 0.
ecc_fail (ECC_FAIL)	19:15	ro	0x0	Pass/fail flag for each ECC block
ecc_value_valid (ECC_VALID)	14:10	ro	0x0	Valid flag for each ECC block.
ecc_read_not_write (ECC_READ_NOT_WRITE)	9	ro	0x0	ECC calculation type: 0: write 1: read
ecc_last_status (ECC_LAST)	8:7	ro	0x0	Last ECC result is updated after completing the ECC calculation: 00: Completed successfully. 01: Unaligned Address, or out-of-range. 10: Data stop after incomplete block. 11: Data stopped but values not read/written because of ecc_jump value.

Field Name	Bits	Type	Reset Value	Description
ecc_status (ECC_STATUS)	6	ro	0x0	Status of the ECC block: 0: idle 1: busy
raw_int_status (ECC_STATUS_RAW_INTERRUPT_STATUS)	5:0	ro	0x0	The interrupts are: Bit [5] Abort. Bit [4] Extra block (if used). Bit [3] Block 3. Bit [2] Block 2. Bit [1] Block 1. Bit [0] Block 0. To clear the interrupt, write a 1 to the bit.

Register ([p1353](#)) ecc_memcfg_1

Name	ecc_memcfg_1
Relative Address	0x00000404
Absolute Address	0xE000E404
Width	13 bits
Access Type	rw
Reset Value	0x00000043
Description	ECC Memory Config

Register ecc_memcfg_1 Details

The ecc_memcfg Register is read-write and contains information about the structure of the memory. Note; You must not write to this register while the ECC block is busy. You can read the current ECC block status from the ECC Status Register.

Field Name	Bits	Type	Reset Value	Description
ecc_extra_block_size (ECC_MEMCFG_ECC_EXTRA_BLOCK_SIZE)	12:11	rw	0x0	The size of the extra block in memory after the last 512 block: 00: 4 bytes 01: 8 bytes 10: 16 bytes 11: 32 bytes Note: These bits are only present if you configure the SMC to use the ECC Extra Block Enable option.
ecc_extra_block (ECC_MEMCFG_ECC_EXTRA_BLOCK)	10	rw	0x0	If configured, this enables a small block for extra information after the last 512 bytes block in the page. Note: These bits are only present if the ECC Extra Block Enable option is configured. 0: 1:
ecc_int_abort (ECC_MEMCFG_ECC_INT_ABORT)	9	rw	0x0	Interrupt on ECC abort: 0: don't assert 1: assert
ecc_int_pass (ECC_MEMCFG_ECC_INT_PASS)	8	rw	0x0	Interrupt when a correct ECC value is read from memory: 0: don't assert 1: assert
ecc_ignore_add_eight (ECC_MEMCFG_IGNORE_ADD8)	7	rw	0x0	Use to indicate if A8 is output with the address, required to find the aligned start of blocks: 0: A8 is output 1: A8 is not output
ecc_jump (ECC_MEMCFG_ECC_JUMP)	6:5	rw	0x2	Indicate that the memory supports column change address commands: 00: no jumping, reads and writes only occur at end of page 01: jump using column change commands 10: jump using full command 11: reserved
ecc_read_end (ECC_MEMCFG_ECC_READ_END)	4	rw	0x0	Indicate when ECC values are read from memory: 0: ECC value for a block must be read immediately after the block. Data access must stop on a 512 byte boundary. 1: ECC values for all blocks are read at the end of the page.

Field Name	Bits	Type	Reset Value	Description
ecc_mode (ECC_MEMCFG_ECC_MODE)	3:2	rw	0x0	Specify the mode of the ECC block: 00: bypassed 01: ECC values are calculated and made available on the APB interface. But they are not read to or written from memory. 10: ECC values and calculated and read/written to memory. For a read, the ECC value is checked and the result of the check is made available on the APB interface. 11: reserved
page_size (ECC_MEMCFG_PAG_E_SIZE)	1:0	rw	0x3	The number of 512 byte blocks in a page: 00: No 512 byte blocks. Reserved if an ecc_extra_block is not configured and enabled. 01: One 512 byte block. 10: Two 512 byte blocks. 11: Four 512 byte blocks.

Register ([p1353](#)) ecc_memcommand1_1

Name	ecc_memcommand1_1
Relative Address	0x00000408
Absolute Address	0xE000E408
Width	25 bits
Access Type	rw
Reset Value	0x01300080
Description	Commands used to detect start of ECC operation

Register ecc_memcommand1_1 Details

The ecc_memcommand1

is read-write and contains the commands that the ECC block uses to detect the start of an ECC operation.

Field Name	Bits	Type	Reset Value	Description
nand_rd_cmd_end_val id (ECC_MEMCOMMAND1_RD_CMD_END_VALID)	24	rw	0x1	Use the end command: 0: 1:
nand_rd_cmd_end (ECC_MEMCOMMAND1_RD_CMD_END)	23:16	rw	0x30	Use the NAND command to initiate a write (0x30).

Field Name	Bits	Type	Reset Value	Description
nand_rd_cmd (ECC_MEMCOMMAN D1_RD_CMD)	15:8	rw	0x0	Use the NAND command used to initiate a read (0x00).
nand_wr_cmd (ECC_MEMCOMMAN D1_WR_CMD)	7:0	rw	0x80	Use the NAND command to initiate a write (0x80).

Register ([p1353](#)) ecc_memcommand2_1

Name	ecc_memcommand2_1
Relative Address	0x0000040C
Absolute Address	0xE000E40C
Width	25 bits
Access Type	rw
Reset Value	0x01E00585
Description	Commands used to access memory within a page

Register ecc_memcommand2_1 Details

The ecc_memcommand2 Register is read-write and contains the commands that the ECC block uses to access different parts of a NAND page. The reset value is suitable for ONFI 1.0 compliant devices

Field Name	Bits	Type	Reset Value	Description
nand_rd_col_change_e nd_valid (ECC_MEMCOMMAN D2_RD_COL_CHANG E_END_VALID)	24	rw	0x1	Use the end command: 0: 1:
nand_rd_col_change_e nd (ECC_MEMCOMMAN D2_RD_COL_CHANG E_END)	23:16	rw	0xE0	Use the NAND command to initiate a write.
nand_rd_col_change (ECC_MEMCOMMAN D2_RD_COL_CHANG E)	15:8	rw	0x5	Use the NAND command to initiate a read or Spare bits pointer command.
nand_wr_col_change (ECC_MEMCOMMAN D2_WR_COL_CHAN GE)	7:0	rw	0x85	The NAND command used to initiate a write

Register ([p1353](#)) ecc_addr0_1

Name	ecc_addr0_1
Relative Address	0x00000410
Absolute Address	0xE000E410
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Lower 32 bits of ECC address

Register ecc_addr0_1 Details

The ecc_addr0 Register is read-only and contains the lower 32 bits of the ECC address

Field Name	Bits	Type	Reset Value	Description
ecc_addr	31:0	ro	0x0	Address bits 31 to 0

Register ([p1353](#)) ecc_addr1_1

Name	ecc_addr1_1
Relative Address	0x00000414
Absolute Address	0xE000E414
Width	24 bits
Access Type	ro
Reset Value	0x00000000
Description	Upper 24 bits of ECC address

Register ecc_addr1_1 Details

The ecc_addr1 Register is read-only and contains the upper 24 bits of the ECC address.

Field Name	Bits	Type	Reset Value	Description
ecc_addr	23:0	ro	0x0	Address bits 55 to 32

Register ([p1353](#)) ecc_value0_1

Name	ecc_value0_1
Relative Address	0x00000418
Absolute Address	0xE000E418
Width	32 bits
Access Type	ro

Reset Value 0x00000000

Description The five ecc_value Registers are read-only and contain block information for the ECC. Note: Writing any value to an ecc_value Register clears the ecc_int bit.

Register ecc_value0_1 Details

Field Name	Bits	Type	Reset Value	Description
ecc_int (ECC_VALUE_INT)	31	ro	0x0	Interrupt flag for this value: 0: 1:
ecc_valid (ECC_VALUE_VALID)	30	ro	0x0	Indicate if this value is valid: 0: 1:
ecc_read (ECC_VALUE_READ)	29	ro	0x0	Indicate if the ECC value has been read from memory: 0: 1:
ecc_fail (ECC_VALUE_FAIL)	28	ro	0x0	Indicate if this value has failed: 0: 1:
ecc_correct (ECC_VALUE_CORRECT)	27	ro	0x0	Indicate if this block is correctable: 0: 1:
reserved	26:24	ro	0x0	Reserved, read undefined
ecc_value (ECC_VALUE)	23:0	ro	0x0	ECC value of check result for block, depending on ECC configuration

Register ([p1353](#)) ecc_value1_1

Name ecc_value1_1

Relative Address 0x0000041C

Absolute Address 0xE000E41C

Width 32 bits

Access Type ro

Reset Value 0x00000000

Description The five ecc_value Registers are read-only and contain block information for the ECC. Note: writing any value to an ecc_value Register clears the ecc_int bit.

Register ecc_value1_1 Details

Field Name	Bits	Type	Reset Value	Description
ecc_int (ECC_VALUE_INT)	31	ro	0x0	Interrupt flag for this value: 0: 1:
ecc_valid (ECC_VALUE_VALID)	30	ro	0x0	Indicate if this value is valid: 0: 1:
ecc_read (ECC_VALUE_READ)	29	ro	0x0	Indicate if the ECC value has been read from memory: 0: 1:
ecc_fail (ECC_VALUE_FAIL)	28	ro	0x0	Indicate if this value has failed: 0: 1:
ecc_correct (ECC_VALUE_CORRECT)	27	ro	0x0	Indicate if this block is correctable: 0: 1:
reserved	26:24	ro	0x0	Reserved, read undefined
ecc_value (ECC_VALUE)	23:0	ro	0x0	ECC value of check result for block, depending on ECC configuration

Register ([p1353](#)) ecc_value2_1

Name	ecc_value2_1
Relative Address	0x00000420
Absolute Address	0xE000E420
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	<p>The five ecc_value Registers are read-only and contain block information for the ECC.</p> <p>Note: writing any value to an ecc_value Register clears the ecc_int bit.</p>

Register ecc_value2_1 Details

Field Name	Bits	Type	Reset Value	Description
ecc_int (ECC_VALUE_INT)	31	ro	0x0	Interrupt flag for this value: 0: 1:
ecc_valid (ECC_VALUE_VALID)	30	ro	0x0	Indicate if this value is valid: 0: 1:
ecc_read (ECC_VALUE_READ)	29	ro	0x0	Indicate if the ECC value has been read from memory: 0: 1:
ecc_fail (ECC_VALUE_FAIL)	28	ro	0x0	Indicate if this value has failed: 0: 1:
ecc_correct (ECC_VALUE_CORRECT)	27	ro	0x0	Indicate if this block is correctable: 0: 1:
reserved	26:24	ro	0x0	Reserved, read undefined
ecc_value (ECC_VALUE)	23:0	ro	0x0	ECC value of check result for block, depending on ECC configuration

Register ([p1353](#)) ecc_value3_1

Name	ecc_value3_1
Relative Address	0x00000424
Absolute Address	0xE000E424
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	The five ecc_value Registers are read-only and contain block information for the ECC. Note: writing any value to an ecc_value Register clears the ecc_int bit.

Register ecc_value3_1 Details

Field Name	Bits	Type	Reset Value	Description
ecc_int (ECC_VALUE_INT)	31	ro	0x0	Interrupt flag for this value: 0: 1:
ecc_valid (ECC_VALUE_VALID)	30	ro	0x0	Indicate if this value is valid: 0: 1:
ecc_read (ECC_VALUE_READ)	29	ro	0x0	Indicate if the ECC value has been read from memory: 0: 1:
ecc_fail (ECC_VALUE_FAIL)	28	ro	0x0	Indicate if this value has failed: 0: 1:
ecc_correct (ECC_VALUE_CORRECT)	27	ro	0x0	Indicate if this block is correctable: 0: 1:
reserved	26:24	ro	0x0	Reserved, read undefined
ecc_value (ECC_VALUE)	23:0	ro	0x0	ECC value of check result for block, depending on ECC configuration

B.30 SPI Controller (SPI)

Module Name	SPI Controller (SPI)
Software Name	XSPIPS
Base Address	0xE0006000 spi0 0xE0007000 spi1
Description	Serial Peripheral Interface Instance no. 0.
Version	1.0
Doc Version	1.2
Vendor Info	Cadence UART

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
Config_reg0	0x00000000	32	mixed	0x00020000	SPI configuration register
Intr_status_reg0	0x00000004	32	ro	0x00000004	SPI interrupt status register
Intrpt_en_reg0	0x00000008	32	mixed	0x00000000	Interrupt Enable register
Intrpt_dis_reg0	0x0000000C	32	mixed	0x00000000	Interrupt disable register
Intrpt_mask_reg0	0x00000010	32	ro	0x00000000	Interrupt mask register
En_reg0	0x00000014	32	mixed	0x00000000	SPI_Enable Register
Delay_reg0	0x00000018	32	rw	0x00000000	Delay Register
Tx_data_reg0	0x0000001C	32	wo	0x00000000	Transmit Data Register
Rx_data_reg0	0x00000020	32	ro	0x00000000	Receive Data Register
Slave_Idle_count_reg0	0x00000024	32	mixed	0x000000FF	Slave Idle Count Register
TX_thres_reg0	0x00000028	32	rw	0x00000001	TX_FIFO Threshold Register
RX_thres_reg0	0x0000002C	32	rw	0x00000001	RX FIFO Threshold Register
Mod_id_reg0	0x000000FC	32	ro	0x00090106	Module ID register

Register ([SPI](#)) Config_reg0

Name	Config_reg0
Software Name	CR
Relative Address	0x00000000

Absolute Address spi0: 0xE0006000
 spi1: 0xE0007000
 Width 32 bits
 Access Type mixed
 Reset Value 0x00020000
 Description SPI configuration register

Register Config_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:18	ro	0x0	Reserved, read as zero, ignored on write.
Modefail_gen_en	17	rw	0x1	ModeFail Generation Enable 1 = enable 0 = disable
Man_start_com (MANSTRT)	16	wo	0x0	Manual Start Command 1 = start transmission of data 0 = don't care
Man_start_en	15	rw	0x0	Manual Start Enable 1 = enables manual start 0 = auto mode
Manual_CS	14	rw	0x0	Manual CS 1 = manual CS mode 0 = auto mode
CS	13:10	rw	0x0	Peripheral chip select lines xxx0 - slave 0 selected xx01 - slave 1 selected x011 - slave 2 selected 0111 - reserved 1111 - No slave selected
PERI_SEL	9	rw	0x0	Peripheral select decode 1 = allow external 3-to-8 decode 0 = only 1 of 3 selects
REF_CLK	8	rw	0x0	Master reference clock select 1 = not supported 0 = use SPI REFERENCE CLOCK
FIFO_WIDTH	7:6	rw	0x0	FIFO width 00 = 8bits 01 = 16bits 10 = 24bits 11 = 32bits

Field Name	Bits	Type	Reset Value	Description
BAUD_RATE_DIV	5:3	rw	0x0	Master mode baud rate divisor controls the amount the spi_ref_clk is divided inside the SPI block 000 = not supported 001 = divide by 4 010 = divide by 8 011 = divide by 16 100 = divide by 32 101 = divide by 64 110 = divide by 128 111 = divide by 256
CLK_PH (CPHA)	2	rw	0x0	Clock phase 1 = the SPI clock is inactive outside the word 0 = the SPI clock is active outside the word
CLK_POL (CPOL)	1	rw	0x0	Clock polarity outside SPI word 1 = the SPI clock is quiescent high 0 = the SPI clock is quiescent low
MODE_SEL (MSTREN)	0	rw	0x0	Mode select 1 = the SPI is in master mode 0 = the SPI is in slave mode

Register ([SPI](#)) Intr_status_reg0

Name	Intr_status_reg0
Software Name	SR
Relative Address	0x00000004
Absolute Address	spi0: 0xE0006004 spi1: 0xE0007004
Width	32 bits
Access Type	ro
Reset Value	0x00000004
Description	SPI interrupt status register

Register Intr_status_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:7	ro	0x0	Reserved, read as zero, ignored on write.
TX_FIFO_underflow (IXR_TXUF)	6	ro	0x0	TX FIFO underflow, write one to this bit location to clear. 1 = underflow is detected 0 = no underflow has been detected
RX_FIFO_full (IXR_RXFULL)	5	ro	0x0	RX FIFO full 1 = FIFO is full 0 = FIFO is not full
RX_FIFO_not_empty (IXR_RXNEMPTY)	4	ro	0x0	RX FIFO not empty 1 = FIFO has more than or equal to THRESHOLD entries 0 = FIFO has less than RX THRESHOLD entries
TX_FIFO_full (IXR_TXFULL)	3	ro	0x0	TX FIFO full 1 = FIFO is full 0 = FIFO is not full
TX_FIFO_not_full (IXR_TXOW)	2	ro	0x1	TX FIFO not full 1 = FIFO has less than THRESHOLD entries 0 = FIFO has more than or equal to THRESHOLD entries
MODE_FAIL (IXR_MODF)	1	ro	0x0	ModeFail interrupt, write one to this bit location to clear. 1 = a mode fault has occurred 0 = no mode fault has been detected
RX_OVERFLOW (IXR_RXOVR)	0	ro	0x0	Receive Overflow interrupt, write one to this bit location to clear. 1 = overflow occurred 0 = no overflow occurred

Register (SPI) Intrpt_en_reg0

Name	Intrpt_en_reg0
Software Name	IER
Relative Address	0x00000008
Absolute Address	spi0: 0xE0006008 spi1: 0xE0007008
Width	32 bits
Access Type	mixed
Reset Value	0x00000000

Description Interrupt Enable register

Register Intrpt_en_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:7	ro	0x0	Reserved, read as zero, ignored on write.
TX_FIFO_underflow (IXR_TXUF)	6	wo	0x0	TX FIFO underflow enable 1 = enable the interrupt 0 = no effect
RX_FIFO_full (IXR_RXFULL)	5	wo	0x0	RX FIFO full enable 1 = enable the interrupt 0 = no effect
RX_FIFO_not_empty (IXR_RXNEMPTY)	4	wo	0x0	RX FIFO not empty enable 1 = enable the interrupt 0 = no effect
TX_FIFO_full (IXR_TXFULL)	3	wo	0x0	TX FIFO full enable 1 = enable the interrupt 0 = no effect
TX_FIFO_not_full (IXR_TXOW)	2	wo	0x0	TX FIFO not full enable 1 = enable the interrupt 0 = no effect
MODE_FAIL (IXR_MODF)	1	wo	0x0	ModeFail interrupt enable 1 = enable the interrupt 0 = no effect
RX_OVERFLOW (IXR_RXOVR)	0	wo	0x0	Receive Overflow interrupt enable 1 = enable the interrupt 0 = no effect

Register ([SPI](#)) Intrpt_dis_reg0

Name Intrpt_dis_reg0

Software Name IDR

Relative Address 0x0000000C

Absolute Address spi0: 0xE000600C
spi1: 0xE000700C

Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Interrupt disable register

Register Intrpt_dis_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:7	ro	0x0	Reserved, read as zero, ignored on write.
TX_FIFO_underflow (IXR_TXUF)	6	wo	0x0	TX FIFO underflow enable 1 = disables the interrupt 0 = no effect
RX_FIFO_full (IXR_RXFULL)	5	wo	0x0	RX FIFO full enable 1 = disables the interrupt 0 = no effect
RX_FIFO_not_empty (IXR_RXNEMPTY)	4	wo	0x0	RX FIFO not empty enable 1 = disables the interrupt 0 = no effect
TX_FIFO_full (IXR_TXFULL)	3	wo	0x0	TX FIFO full enable 1 = disables the interrupt 0 = no effect
TX_FIFO_not_full (IXR_TXOW)	2	wo	0x0	TX FIFO not full enable 1 = disables the interrupt 0 = no effect
MODE_FAIL (IXR_MODF)	1	wo	0x0	ModeFail interrupt enable 1 = disables the interrupt 0 = no effect
RX_OVERFLOW (IXR_RXOVR)	0	wo	0x0	Receive Overflow interrupt enable 1 = disables the interrupt 0 = no effect

Register ([SPI](#)) Intrpt_mask_reg0

Name	Intrpt_mask_reg0
Software Name	IMR

Relative Address	0x00000010
Absolute Address	spi0: 0xE0006010 spi1: 0xE0007010
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Interrupt mask register

Register Intrpt_mask_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:7	ro	0x0	Reserved, read as zero, ignored on write.
TX_FIFO_underflow (IXR_TXUF)	6	ro	0x0	TX FIFO underflow enable 1 = interrupt is disabled 0 = interrupt is enabled
RX_FIFO_full (IXR_RXFULL)	5	ro	0x0	RX FIFO full enable 1 = interrupt is disabled 0 = interrupt is enabled
RX_FIFO_not_empty (IXR_RXNEMPTY)	4	ro	0x0	RX FIFO not empty enable 1 = interrupt is disabled 0 = interrupt is enabled
TX_FIFO_full (IXR_TXFULL)	3	ro	0x0	TX FIFO full enable 1 = interrupt is disabled 0 = interrupt is enabled
TX_FIFO_not_full (IXR_TXOW)	2	ro	0x0	TX FIFO not full enable 1 = interrupt is disabled 0 = interrupt is enabled
MODE_FAIL (IXR_MODF)	1	ro	0x0	ModeFail interrupt enable 1 = interrupt is disabled 0 = interrupt is enabled
RX_OVERFLOW (IXR_RXOVR)	0	ro	0x0	Receive Overflow interrupt enable 1 = interrupt is disabled 0 = interrupt is enabled

Register (SPI) En_reg0

Name	En_reg0
Software Name	ER
Relative Address	0x00000014
Absolute Address	spi0: 0xE0006014 spi1: 0xE0007014
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	SPI_Enable Register

Register En_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:1	ro	0x0	Reserved, read as zero, ignored on write.
SPI_EN (ENABLE)	0	rw	0x0	SPI_Enable 1 = enable the SPI 0 = disable the SPI

Register (SPI) Delay_reg0

Name	Delay_reg0
Software Name	DR
Relative Address	0x00000018
Absolute Address	spi0: 0xE0006018 spi1: 0xE0007018
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Delay Register

Register Delay_reg0 Details

Field Name	Bits	Type	Reset Value	Description
d_nss	31:24	rw	0x0	Delay in SPI REFERENCE CLOCK or ext_clk cycles for the length that the master mode chip select outputs are de-asserted between words when cpha=0.
d_btwn (BTWN)	23:16	rw	0x0	Delay in SPI REFERENCE CLOCK or ext_clk cycles between one chip select being de-activated and the activation of another
d_after (AFTER)	15:8	rw	0x0	Delay in SPI REFERENCE CLOCK or ext_clk cycles between last bit of current word and the first bit of the next word.
d_int (INIT)	7:0	rw	0x0	Added delay in SPI REFERENCE CLOCK or ext_clk cycles between setting n_ss_out low and first bit transfer.

Register (SPI) Tx_data_reg0

Name	Tx_data_reg0
Software Name	TXD
Relative Address	0x0000001C
Absolute Address	spi0: 0xE000601C spi1: 0xE000701C
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	Transmit Data Register

Register Tx_data_reg0 Details

Field Name	Bits	Type	Reset Value	Description
TX_FIFO_data	31:0	wo	0x0	Data to TX FIFO

Register (SPI) Rx_data_reg0

Name	Rx_data_reg0
Software Name	RXD

Relative Address	0x00000020
Absolute Address	spl0: 0xE0006020 spl1: 0xE0007020
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Receive Data Register

Register Rx_data_reg0 Details

Field Name	Bits	Type	Reset Value	Description
RX_FIFO_data	31:0	ro	0x0	Data from TX FIFO

Register (SPI) Slave_Idle_count_reg0

Name	Slave_Idle_count_reg0
Software Name	SICR
Relative Address	0x00000024
Absolute Address	spl0: 0xE0006024 spl1: 0xE0007024
Width	32 bits
Access Type	mixed
Reset Value	0x000000FF
Description	Slave Idle Count Register

Register Slave_Idle_count_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	ro	0x0	Reserved, read as zero, ignored on write.
Slave_Idle_coun	7:0	rw	0xFF	SPI in slave mode detects a start only when the external SPI master serial clock (sclk_in) is stable (quiescent state) for SPI REFERENCE CLOCK cycles specified by slave idle count register or when the SPI is deselected.

Register (SPI) TX_thres_reg0

Name	TX_thres_reg0
------	---------------

Software Name	TXWR
Relative Address	0x00000028
Absolute Address	spi0: 0xE0006028 spi1: 0xE0007028
Width	32 bits
Access Type	rw
Reset Value	0x00000001
Description	TX_FIFO Threshold Register

Register TX_thres_reg0 Details

Field Name	Bits	Type	Reset Value	Description
DEPTH_of_TX_FIFO	31:0	rw	0x1	Defines the level at which the TX FIFO not full interrupt is generated

Register ([SPI](#)) RX_thres_reg0

Name	RX_thres_reg0
Relative Address	0x0000002C
Absolute Address	spi0: 0xE000602C spi1: 0xE000702C
Width	32 bits
Access Type	rw
Reset Value	0x00000001
Description	RX FIFO Threshold Register

Register RX_thres_reg0 Details

Field Name	Bits	Type	Reset Value	Description
DEPTH_of_RX_FIFO	31:0	rw	0x1	Defines the level at which the RX FIFO not empty interrupt is generated

Register ([SPI](#)) Mod_id_reg0

Name	Mod_id_reg0
Relative Address	0x000000FC
Absolute Address	spi0: 0xE00060FC spi1: 0xE00070FC
Width	32 bits

Access Type ro
Reset Value 0x00090106
Description Module ID register

Register Mod_id_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:25	ro	0x0	Reserved, read as zero, ignored on write.
module_ID	24:0	ro	0x90106	Module ID number

B.31 System Watchdog Timer (swdt)

Module Name	System Watchdog Timer (swdt)
Software Name	XWDTPS
Base Address	0xF8005000 swdt
Description	System Watchdog Timer Registers
Version	7.0
Doc Version	2.1
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
MODE	0x00000000	24	mixed	0x000001C2	WD zero mode register
CONTROL	0x00000004	26	mixed	0x03FFC3FC	Counter Control Register
RESTART	0x00000008	16	wo	0x00000000	Restart key register - this not a real register as no data is stored
STATUS	0x0000000C	1	ro	0x00000000	Status Register

Register ([swdt](#)) MODE

Name	MODE
Software Name	ZMR
Relative Address	0x00000000
Absolute Address	0xF8005000
Width	24 bits
Access Type	mixed
Reset Value	0x000001C2
Description	WD zero mode register

Register MODE Details

Field Name	Bits	Type	Reset Value	Description
ZKEY	23:12	wo	0x0	Zero access key - writes to the zero mode register are only valid if this field is set to 0xABC; this field is write only.
reserved	11:9	waz	0x0	Should be zero (sbz)

Field Name	Bits	Type	Reset Value	Description
IRQLN	8:7	rw	0x3	Interrupt request length - selects the number of pclk cycles during which an interrupt request is held active after it is invoked: 00 = 4 01 = 8 10 = 16 11 = 32
RSTLN	6:4	rw	0x4	Reset length - selects the number of clock cycles (pclk) during which the internal system reset is held active after it is invoked: 000 = 2 001 = 4 010 = 8 011 = 16 100 = 32 101 = 64 110 = 128 111 = 256 Note: The minimum number of cycles required for an AMBA reset is two.
reserved	3	waz	0x0	Should be zero (sbz)
IRQEN	2	rw	0x0	Interrupt request enable - if set, the watchdog will issue an interrupt request when the counter reaches zero, if WDEN = 1.
RSTEN	1	rw	0x1	Reset enable - if set, the watchdog will issue an internal reset when the counter reaches zero, if WDEN = 1.
WDEN	0	rw	0x0	Watchdog enable - if set, the watchdog is enabled and can generate any signals that are enabled.

Register ([swdt](#)) CONTROL

Name	CONTROL
Software Name	CCR
Relative Address	0x00000004
Absolute Address	0xF8005004
Width	26 bits
Access Type	mixed
Reset Value	0x03FFC3FC
Description	Counter Control Register

Register CONTROL Details

Field Name	Bits	Type	Reset Value	Description
CKEY	25:14	wo	0xFFF	Counter access key - writes to the control register are only valid if this field is set to 0x248; this field is write only.
CRV	13:2	rw	0xFF	Counter restart value - the counter is restarted with the value 0xNNNFFF, where NNN is the value of this field.
CLKSEL	1:0	rw	0x0	Counter clock prescale - selects the prescaler division ratio: 00 = pclk divided by 8 01 = pclk divided by 64 10 = pclk divided by 256 11 = pclk divided by 4096 Note: If a restart signal is received the prescaler should be reset.

Register ([swdt](#)) RESTART

Name	RESTART
Relative Address	0x00000008
Absolute Address	0xF8005008
Width	16 bits
Access Type	wo
Reset Value	0x00000000
Description	Restart key register - this not a real register as no data is stored

Register RESTART Details

Field Name	Bits	Type	Reset Value	Description
RSTKEY (KEY_VAL)	15:0	wo	0x0	Restart key - the watchdog is restarted if this field is set to the value 0x1999

Register ([swdt](#)) STATUS

Name	STATUS
Software Name	SR
Relative Address	0x0000000C
Absolute Address	0xF800500C
Width	1 bits

Access Type ro
Reset Value 0x00000000
Description Status Register

Register STATUS Details

Field Name	Bits	Type	Reset Value	Description
WDZ	0	ro	0x0	set when the watchdog reaches zero count

B.32 Triple Timer Counter (ttc)

Module Name	Triple Timer Counter (ttc)
Software Name	XTTCPS
Base Address	0xF8001000 ttc0 0xF8002000 ttc1
Description	Triple Timer Counter Instance no. 0.
Version	5.0
Doc Version	2.0
Vendor Info	

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
Clock_Control_1	0x00000000	7	rw	0x00000000	Clock Control register
Clock_Control_2	0x00000004	7	rw	0x00000000	Clock Control register
Clock_Control_3	0x00000008	7	rw	0x00000000	Clock Control register
Counter_Control_1	0x0000000C	7	rw	0x00000021	Operational mode and reset
Counter_Control_2	0x00000010	7	rw	0x00000021	Operational mode and reset
Counter_Control_3	0x00000014	7	rw	0x00000021	Operational mode and reset
Counter_Value_1	0x00000018	16	ro	0x00000000	Current counter value
Counter_Value_2	0x0000001C	16	ro	0x00000000	Current counter value
Counter_Value_3	0x00000020	16	ro	0x00000000	Current counter value
Interval_Counter_1	0x00000024	16	rw	0x00000000	Interval value
Interval_Counter_2	0x00000028	16	rw	0x00000000	Interval value
Interval_Counter_3	0x0000002C	16	rw	0x00000000	Interval value
Match_1_Counter_1	0x00000030	16	rw	0x00000000	Match value
Match_1_Counter_2	0x00000034	16	rw	0x00000000	Match value
Match_1_Counter_3	0x00000038	16	rw	0x00000000	Match value
Match_2_Counter_1	0x0000003C	16	rw	0x00000000	Match value
Match_2_Counter_2	0x00000040	16	rw	0x00000000	Match value
Match_2_Counter_3	0x00000044	16	rw	0x00000000	Match value
Match_3_Counter_1	0x00000048	16	rw	0x00000000	Match value
Match_3_Counter_2	0x0000004C	16	rw	0x00000000	Match value

Register Name	Address	Width	Type	Reset Value	Description
Match_3_Counter_3	0x00000050	16	rw	0x00000000	Match value
Interrupt_Register_1	0x00000054	6	ro	0x00000000	Counter 1 Interval, Match, Overflow and Event interrupts
Interrupt_Register_2	0x00000058	6	ro	0x00000000	Counter 2 Interval, Match, Overflow and Event interrupts
Interrupt_Register_3	0x0000005C	6	ro	0x00000000	Counter 3 Interval, Match, Overflow and Event interrupts
Interrupt_Enable_1	0x00000060	6	rw	0x00000000	ANDed with corresponding Interrupt Register
Interrupt_Enable_2	0x00000064	6	rw	0x00000000	ANDed with corresponding Interrupt Register
Interrupt_Enable_3	0x00000068	6	rw	0x00000000	ANDed with corresponding Interrupt Register
Event_Control_Timer_1	0x0000006C	3	rw	0x00000000	Enable, pulse and overflow
Event_Control_Timer_2	0x00000070	3	rw	0x00000000	Enable, pulse and overflow
Event_Control_Timer_3	0x00000074	3	rw	0x00000000	Enable, pulse and overflow
Event_Register_1	0x00000078	16	ro	0x00000000	pclk cycle count for event
Event_Register_2	0x0000007C	16	ro	0x00000000	pclk cycle count for event
Event_Register_3	0x00000080	16	ro	0x00000000	pclk cycle count for event

Register ([ttc](#)) Clock_Control_1

Name	Clock_Control_1
Software Name	CLK_CNTRL
Relative Address	0x00000000
Absolute Address	ttc0: 0xF8001000 ttc1: 0xF8002000
Width	7 bits
Access Type	rw
Reset Value	0x00000000
Description	Clock Control register

Register Clock_Control_1 Details

Field Name	Bits	Type	Reset Value	Description
Ex_E (EXT_EDGE)	6	rw	0x0	External Clock Edge: when this bit is set and the extend clock is selected, the counter clocks on the negative going edge of the external clock input.
C_Src (SRC)	5	rw	0x0	Clock Source: when this bit is set the counter uses the external clock input, ext_clk; the default clock source is pclk.
PS_V (PS_VAL)	4:1	rw	0x0	Prescale value (N): if prescale is enabled, the count rate is divided by 2N+1 (divided by 2 to 65536)
PS_En (PS_EN)	0	rw	0x0	Prescale enable: when this bit is set the counter, clock source is prescaled; the default clock source is that defined by C_Src.the default

Register ([ttc](#)) Clock_Control_2

Name	Clock_Control_2
Relative Address	0x00000004
Absolute Address	ttc0: 0xF8001004 ttc1: 0xF8002004
Width	7 bits
Access Type	rw
Reset Value	0x00000000
Description	Clock Control register

Register Clock_Control_2 Details

Field Name	Bits	Type	Reset Value	Description
Ex_E (CLK_CNTRL_EXT_EDGE)	6	rw	0x0	External Clock Edge: when this bit is set and the extend clock is selected, the counter clocks on the negative going edge of the external clock input.
C_Src (CLK_CNTRL_SRC)	5	rw	0x0	Clock Source: when this bit is set the counter uses the external clock input, ext_clk; the default clock source is pclk.
PS_V (CLK_CNTRL_PS_VAL)	4:1	rw	0x0	Prescale value (N): if prescale is enabled, the count rate is divided by 2N+1 (divided by 2 to 65536)
PS_En (CLK_CNTRL_PS_EN)	0	rw	0x0	Prescale enable: when this bit is set the counter, clock source is prescaled; the default clock source is that defined by C_Src.the default

Register ([ttc](#)) Clock_Control_3

Name	Clock_Control_3
Relative Address	0x00000008
Absolute Address	ttc0: 0xF8001008 ttc1: 0xF8002008
Width	7 bits
Access Type	rw
Reset Value	0x00000000
Description	Clock Control register

Register Clock_Control_3 Details

Field Name	Bits	Type	Reset Value	Description
Ex_E (CLK_CNTRL_EXT_E DGE)	6	rw	0x0	External Clock Edge: when this bit is set and the extend clock is selected, the counter clocks on the negative going edge of the external clock input.
C_Src (CLK_CNTRL_SRC)	5	rw	0x0	Clock Source: when this bit is set the counter uses the external clock input, ext_clk; the default clock source is pclk.
PS_V (CLK_CNTRL_PS_VA L)	4:1	rw	0x0	Prescale value (N): if prescale is enabled, the count rate is divided by 2N+1 (divided by 2 to 65536)
PS_En (CLK_CNTRL_PS_EN)	0	rw	0x0	Prescale enable: when this bit is set the counter, clock source is prescaled; the default clock source is that defined by C_Src.the default

Register ([ttc](#)) Counter_Control_1

Name	Counter_Control_1
Software Name	CNT_CNTRL
Relative Address	0x0000000C
Absolute Address	ttc0: 0xF800100C ttc1: 0xF800200C
Width	7 bits
Access Type	rw
Reset Value	0x00000021
Description	Operational mode and reset

Register Counter_Control_1 Details

Field Name	Bits	Type	Reset Value	Description
Wave_pol (POL_WAVE)	6	rw	0x0	Waveform polarity: When this bit is high, the waveform output goes from high to low on Match_1 interrupt and returns high on overflow or interval interrupt; when low, the waveform goes from low to high on Match_1 interrupt and returns low on overflow or interval interrupt.
Wave_en (EN_WAVE)	5	rw	0x1	Output waveform enable, active low.
RST	4	rw	0x0	Setting this bit high resets the counter value and restarts counting; the RST bit is automatically cleared on restart.
Match (MATCH)	3	rw	0x0	Register Match mode: when Match is set, an interrupt is generated when the count value matches one of the three match registers and the corresponding bit is set in the Interrupt Enable register.
DEC (DECR)	2	rw	0x0	Decrement: when this bit is high the counter counts down.
INT	1	rw	0x0	When this bit is high, the timer is in Interval Mode, and the counter generates interrupts at regular intervals; when low, the timer is in overflow mode.
DIS	0	rw	0x1	Disable counter: when this bit is high, the counter is stopped, holding its last value until reset, restarted or enabled again.

Register ([ttc](#)) Counter_Control_2

Name	Counter_Control_2
Relative Address	0x00000010
Absolute Address	ttc0: 0xF8001010 ttc1: 0xF8002010
Width	7 bits
Access Type	rw
Reset Value	0x00000021
Description	Operational mode and reset

Register Counter_Control_2 Details

Field Name	Bits	Type	Reset Value	Description
Wave_pol (CNT_CNTRL_POL_WAVE)	6	rw	0x0	Waveform polarity: When this bit is high, the waveform output goes from high to low on Match_1 interrupt and returns high on overflow or interval interrupt; when low, the waveform goes from low to high on Match_1 interrupt and returns low on overflow or interval interrupt.
Wave_en (CNT_CNTRL_EN_WAVE)	5	rw	0x1	Output waveform enable, active low.
RST (CNT_CNTRL_RST)	4	rw	0x0	Setting this bit high resets the counter value and restarts counting; the RST bit is automatically cleared on restart.
Match (CNT_CNTRL_MATCH)	3	rw	0x0	Register Match mode: when Match is set, an interrupt is generated when the count value matches one of the three match registers and the corresponding bit is set in the Interrupt Enable register.
DEC (CNT_CNTRL_DECR)	2	rw	0x0	Decrement: when this bit is high the counter counts down.
INT (CNT_CNTRL_INT)	1	rw	0x0	When this bit is high, the timer is in Interval Mode, and the counter generates interrupts at regular intervals; when low, the timer is in overflow mode.
DIS (CNT_CNTRL_DIS)	0	rw	0x1	Disable counter: when this bit is high, the counter is stopped, holding its last value until reset, restarted or enabled again.

Register ([ttc](#)) Counter_Control_3

Name	Counter_Control_3
Relative Address	0x00000014
Absolute Address	ttc0: 0xF8001014 ttc1: 0xF8002014
Width	7 bits
Access Type	rw
Reset Value	0x00000021
Description	Operational mode and reset

Register Counter_Control_3 Details

Field Name	Bits	Type	Reset Value	Description
Wave_pol (CNT_CNTRL_POL_WAVE)	6	rw	0x0	Waveform polarity: When this bit is high, the waveform output goes from high to low on Match_1 interrupt and returns high on overflow or interval interrupt; when low, the waveform goes from low to high on Match_1 interrupt and returns low on overflow or interval interrupt.
Wave_en (CNT_CNTRL_EN_WAVE)	5	rw	0x1	Output waveform enable, active low.
RST (CNT_CNTRL_RST)	4	rw	0x0	Setting this bit high resets the counter value and restarts counting; the RST bit is automatically cleared on restart.
Match (CNT_CNTRL_MATCH)	3	rw	0x0	Register Match mode: when Match is set, an interrupt is generated when the count value matches one of the three match registers and the corresponding bit is set in the Interrupt Enable register.
DEC (CNT_CNTRL_DECR)	2	rw	0x0	Decrement: when this bit is high the counter counts down.
INT (CNT_CNTRL_INT)	1	rw	0x0	When this bit is high, the timer is in Interval Mode, and the counter generates interrupts at regular intervals; when low, the timer is in overflow mode.
DIS (CNT_CNTRL_DIS)	0	rw	0x1	Disable counter: when this bit is high, the counter is stopped, holding its last value until reset, restarted or enabled again.

Register ([ttc](#)) Counter_Value_1

Name	Counter_Value_1
Software Name	COUNT_VALUE
Relative Address	0x00000018
Absolute Address	ttc0: 0xF8001018 ttc1: 0xF8002018
Width	16 bits
Access Type	ro
Reset Value	0x00000000
Description	Current counter value

Register Counter_Value_1 Details

Field Name	Bits	Type	Reset Value	Description
Value (MASK)	15:0	ro	0x0	At any time, a Timer Counter's count value can be read from its Counter Value Register.

Register ([ttc](#)) Counter_Value_2

Name	Counter_Value_2
Relative Address	0x0000001C
Absolute Address	ttc0: 0xF800101C ttc1: 0xF800201C
Width	16 bits
Access Type	ro
Reset Value	0x00000000
Description	Current counter value

Register Counter_Value_2 Details

Field Name	Bits	Type	Reset Value	Description
Value (COUNT_VALUE)	15:0	ro	0x0	At any time, a Timer Counter's count value can be read from its Counter Value Register.

Register ([ttc](#)) Counter_Value_3

Name	Counter_Value_3
Relative Address	0x00000020
Absolute Address	ttc0: 0xF8001020 ttc1: 0xF8002020
Width	16 bits
Access Type	ro
Reset Value	0x00000000
Description	Current counter value

Register Counter_Value_3 Details

Field Name	Bits	Type	Reset Value	Description
Value (COUNT_VALUE)	15:0	ro	0x0	At any time, a Timer Counter's count value can be read from its Counter Value Register.

Register ([ttc](#)) Interval_Counter_1

Name	Interval_Counter_1
Software Name	INTERVAL_VAL
Relative Address	0x00000024
Absolute Address	ttc0: 0xF8001024 ttc1: 0xF8002024
Width	16 bits
Access Type	rw
Reset Value	0x00000000
Description	Interval value

Register Interval_Counter_1 Details

Field Name	Bits	Type	Reset Value	Description
Interval (COUNT_VALUE)	15:0	rw	0x0	If interval is enabled, this is the maximum value that the counter will count up to or down from.

Register ([ttc](#)) Interval_Counter_2

Name	Interval_Counter_2
Relative Address	0x00000028
Absolute Address	ttc0: 0xF8001028 ttc1: 0xF8002028
Width	16 bits
Access Type	rw
Reset Value	0x00000000
Description	Interval value

Register Interval_Counter_2 Details

Field Name	Bits	Type	Reset Value	Description
Interval (INTERVAL_VAL)	15:0	rw	0x0	If interval is enabled, this is the maximum value that the counter will count up to or down from.

Register ([ttc](#)) Interval_Counter_3

Name	Interval_Counter_3
Relative Address	0x0000002C

Absolute Address	ttc0: 0xF800102C ttc1: 0xF800202C
Width	16 bits
Access Type	rw
Reset Value	0x00000000
Description	Interval value

Register Interval_Counter_3 Details

Field Name	Bits	Type	Reset Value	Description
Interval (INTERVAL_VAL)	15:0	rw	0x0	If interval is enabled, this is the maximum value that the counter will count up to or down from.

Register ([ttc](#)) Match_1_Counter_1

Name	Match_1_Counter_1
Software Name	MATCH_0
Relative Address	0x00000030
Absolute Address	ttc0: 0xF8001030 ttc1: 0xF8002030
Width	16 bits
Access Type	rw
Reset Value	0x00000000
Description	Match value

Register Match_1_Counter_1 Details

Field Name	Bits	Type	Reset Value	Description
Match (MATCH)	15:0	rw	0x0	When a counter has the same value as is stored in one of its match registers and match mode is enabled, a match interrupt is generated. Each counter has three match registers.

Register ([ttc](#)) Match_1_Counter_2

Name	Match_1_Counter_2
Relative Address	0x00000034
Absolute Address	ttc0: 0xF8001034 ttc1: 0xF8002034
Width	16 bits

Access Type	rw
Reset Value	0x00000000
Description	Match value

Register Match_1_Counter_2 Details

Field Name	Bits	Type	Reset Value	Description
Match (MATCH)	15:0	rw	0x0	When a counter has the same value as is stored in one of its match registers and match mode is enabled, a match interrupt is generated. Each counter has three match registers.

Register ([ttc](#)) Match_1_Counter_3

Name	Match_1_Counter_3
Relative Address	0x00000038
Absolute Address	ttc0: 0xF8001038 ttc1: 0xF8002038
Width	16 bits
Access Type	rw
Reset Value	0x00000000
Description	Match value

Register Match_1_Counter_3 Details

Field Name	Bits	Type	Reset Value	Description
Match (MATCH)	15:0	rw	0x0	When a counter has the same value as is stored in one of its match registers and match mode is enabled, a match interrupt is generated. Each counter has three match registers.

Register ([ttc](#)) Match_2_Counter_1

Name	Match_2_Counter_1
Software Name	MATCH_1
Relative Address	0x0000003C
Absolute Address	ttc0: 0xF800103C ttc1: 0xF800203C
Width	16 bits
Access Type	rw
Reset Value	0x00000000

Description Match value

Register Match_2_Counter_1 Details

Field Name	Bits	Type	Reset Value	Description
Match (MATCH)	15:0	rw	0x0	When a counter has the same value as is stored in one of its match registers and match mode is enabled, a match interrupt is generated. Each counter has three match registers.

Register ([ttc](#)) Match_2_Counter_2

Name Match_2_Counter_2

Relative Address 0x00000040

Absolute Address ttc0: 0xF8001040
ttc1: 0xF8002040

Width 16 bits

Access Type rw

Reset Value 0x00000000

Description Match value

Register Match_2_Counter_2 Details

Field Name	Bits	Type	Reset Value	Description
Match (MATCH)	15:0	rw	0x0	When a counter has the same value as is stored in one of its match registers and match mode is enabled, a match interrupt is generated. Each counter has three match registers.

Register ([ttc](#)) Match_2_Counter_3

Name Match_2_Counter_3

Relative Address 0x00000044

Absolute Address ttc0: 0xF8001044
ttc1: 0xF8002044

Width 16 bits

Access Type rw

Reset Value 0x00000000

Description Match value

Register Match_2_Counter_3 Details

Field Name	Bits	Type	Reset Value	Description
Match (MATCH)	15:0	rw	0x0	When a counter has the same value as is stored in one of its match registers and match mode is enabled, a match interrupt is generated. Each counter has three match registers.

Register ([ttc](#)) Match_3_Counter_1

Name	Match_3_Counter_1
Software Name	MATCH_2
Relative Address	0x00000048
Absolute Address	ttc0: 0xF8001048 ttc1: 0xF8002048
Width	16 bits
Access Type	rw
Reset Value	0x00000000
Description	Match value

Register Match_3_Counter_1 Details

Field Name	Bits	Type	Reset Value	Description
Match (MATCH)	15:0	rw	0x0	When a counter has the same value as is stored in one of its match registers and match mode is enabled, a match interrupt is generated. Each counter has three match registers.

Register ([ttc](#)) Match_3_Counter_2

Name	Match_3_Counter_2
Relative Address	0x0000004C
Absolute Address	ttc0: 0xF800104C ttc1: 0xF800204C
Width	16 bits
Access Type	rw
Reset Value	0x00000000
Description	Match value

Register Match_3_Counter_2 Details

Field Name	Bits	Type	Reset Value	Description
Match (MATCH)	15:0	rw	0x0	When a counter has the same value as is stored in one of its match registers and match mode is enabled, a match interrupt is generated. Each counter has three match registers.

Register ([ttc](#)) Match_3_Counter_3

Name	Match_3_Counter_3
Relative Address	0x00000050
Absolute Address	ttc0: 0xF8001050 ttc1: 0xF8002050
Width	16 bits
Access Type	rw
Reset Value	0x00000000
Description	Match value

Register Match_3_Counter_3 Details

Field Name	Bits	Type	Reset Value	Description
Match (MATCH)	15:0	rw	0x0	When a counter has the same value as is stored in one of its match registers and match mode is enabled, a match interrupt is generated. Each counter has three match registers.

Register ([ttc](#)) Interrupt_Register_1

Name	Interrupt_Register_1
Software Name	ISR
Relative Address	0x00000054
Absolute Address	ttc0: 0xF8001054 ttc1: 0xF8002054
Width	6 bits
Access Type	ro
Reset Value	0x00000000
Description	Counter 1 Interval, Match, Overflow and Event interrupts

Register Interrupt_Register_1 Details

Field Name	Bits	Type	Reset Value	Description
Ev	5	ro	0x0	Event timer overflow interrupt
Ov (IXR_CNT_OVR)	4	ro	0x0	Counter overflow
M3 (IXR_MATCH_2)	3	ro	0x0	Match 3 interrupt
M2 (IXR_MATCH_1)	2	ro	0x0	Match 2 interrupt
M1 (IXR_MATCH_0)	1	ro	0x0	Match 1 interrupt
Iv (IXR_INTERVAL)	0	ro	0x0	Interval interrupt

Register ([ttc](#)) Interrupt_Register_2

Name	Interrupt_Register_2
Relative Address	0x00000058
Absolute Address	ttc0: 0xF8001058 ttc1: 0xF8002058
Width	6 bits
Access Type	ro
Reset Value	0x00000000
Description	Counter 2 Interval, Match, Overflow and Event interrupts

Register Interrupt_Register_2 Details

Field Name	Bits	Type	Reset Value	Description
Ev	5	ro	0x0	Event timer overflow interrupt
Ov (IXR_CNT_OVR)	4	ro	0x0	Counter overflow
M3 (IXR_MATCH_2)	3	ro	0x0	Match 3 interrupt
M2 (IXR_MATCH_1)	2	ro	0x0	Match 2 interrupt
M1 (IXR_MATCH_0)	1	ro	0x0	Match 1 interrupt
Iv (IXR_INTERVAL)	0	ro	0x0	Interval interrupt

Register ([ttc](#)) Interrupt_Register_3

Name	Interrupt_Register_3
Relative Address	0x0000005C
Absolute Address	ttc0: 0xF800105C ttc1: 0xF800205C
Width	6 bits
Access Type	ro
Reset Value	0x00000000
Description	Counter 3 Interval, Match, Overflow and Event interrupts

Register Interrupt_Register_3 Details

Field Name	Bits	Type	Reset Value	Description
Ev	5	ro	0x0	Event timer overflow interrupt
Ov (IXR_CNT_OVR)	4	ro	0x0	Counter overflow
M3 (IXR_MATCH_2)	3	ro	0x0	Match 3 interrupt
M2 (IXR_MATCH_1)	2	ro	0x0	Match 2 interrupt
M1 (IXR_MATCH_0)	1	ro	0x0	Match 1 interrupt
Iv (IXR_INTERVAL)	0	ro	0x0	Interval interrupt

Register ([ttc](#)) Interrupt_Enable_1

Name	Interrupt_Enable_1
Software Name	IER
Relative Address	0x00000060
Absolute Address	ttc0: 0xF8001060 ttc1: 0xF8002060
Width	6 bits
Access Type	rw
Reset Value	0x00000000
Description	ANDed with corresponding Interrupt Register

Register Interrupt_Enable_1 Details

Field Name	Bits	Type	Reset Value	Description
IEN	5:0	rw	0x0	Enables for bits 05:00 in Interrupt Register: corresponding bits must be set to enable the interrupt.

Register ([ttc](#)) Interrupt_Enable_2

Name	Interrupt_Enable_2
Relative Address	0x00000064
Absolute Address	ttc0: 0xF8001064 ttc1: 0xF8002064
Width	6 bits
Access Type	rw
Reset Value	0x00000000
Description	ANDed with corresponding Interrupt Register

Register Interrupt_Enable_2 Details

Field Name	Bits	Type	Reset Value	Description
IEN	5:0	rw	0x0	Enables for bits 05:00 in Interrupt Register: corresponding bits must be set to enable the interrupt.

Register ([ttc](#)) Interrupt_Enable_3

Name	Interrupt_Enable_3
Relative Address	0x00000068
Absolute Address	ttc0: 0xF8001068 ttc1: 0xF8002068
Width	6 bits
Access Type	rw
Reset Value	0x00000000
Description	ANDed with corresponding Interrupt Register

Register Interrupt_Enable_3 Details

Field Name	Bits	Type	Reset Value	Description
IEN	5:0	rw	0x0	Enables for bits 05:00 in Interrupt Register: corresponding bits must be set to enable the interrupt.

Register ([ttc](#)) Event_Control_Timer_1

Name	Event_Control_Timer_1
Relative Address	0x0000006C
Absolute Address	ttc0: 0xF800106C ttc1: 0xF800206C
Width	3 bits
Access Type	rw
Reset Value	0x00000000
Description	Enable, pulse and overflow

Register Event_Control_Timer_1 Details

Field Name	Bits	Type	Reset Value	Description
E_Ov	2	rw	0x0	When this bit is low, the event timer is disabled and set to zero when an Event Timer Register overflow occurs; when set high, the timer continues counting on overflow.
E_Lo	1	rw	0x0	When this bit is high, the timer counts pclk cycles during the low level duration of ext_clk; when low, the event timer counts the high level duration of ext_clk.
E_En	0	rw	0x0	Enable timer: when this bit is high, the event timer is enabled.

Register ([ttc](#)) Event_Control_Timer_2

Name	Event_Control_Timer_2
Relative Address	0x00000070
Absolute Address	ttc0: 0xF8001070 ttc1: 0xF8002070
Width	3 bits
Access Type	rw
Reset Value	0x00000000
Description	Enable, pulse and overflow

Register Event_Control_Timer_2 Details

Field Name	Bits	Type	Reset Value	Description
E_Ov	2	rw	0x0	When this bit is low, the event timer is disabled and set to zero when an Event Timer Register overflow occurs; when set high, the timer continues counting on overflow.
E_Lo	1	rw	0x0	When this bit is high, the timer counts pclk cycles during the low level duration of ext_clk; when low, the event timer counts the high level duration of ext_clk.
E_En	0	rw	0x0	Enable timer: when this bit is high, the event timer is enabled.

Register ([ttc](#)) Event_Control_Timer_3

Name	Event_Control_Timer_3
Relative Address	0x00000074
Absolute Address	ttc0: 0xF8001074 ttc1: 0xF8002074
Width	3 bits
Access Type	rw
Reset Value	0x00000000
Description	Enable, pulse and overflow

Register Event_Control_Timer_3 Details

Field Name	Bits	Type	Reset Value	Description
E_Ov	2	rw	0x0	When this bit is low, the event timer is disabled and set to zero when an Event Timer Register overflow occurs; when set high, the timer continues counting on overflow.
E_Lo	1	rw	0x0	When this bit is high, the timer counts pclk cycles during the low level duration of ext_clk; when low, the event timer counts the high level duration of ext_clk.
E_En	0	rw	0x0	Enable timer: when this bit is high, the event timer is enabled.

Register ([ttc](#)) Event_Register_1

Name	Event_Register_1
Relative Address	0x00000078

Absolute Address	ttc0: 0xF8001078 ttc1: 0xF8002078
Width	16 bits
Access Type	ro
Reset Value	0x00000000
Description	pclk cycle count for event

Register Event_Register_1 Details

Field Name	Bits	Type	Reset Value	Description
Event	15:0	ro	0x0	This register stores the result of the pclk count during the ext_clk high or low pulse.

Register ([ttc](#)) Event_Register_2

Name	Event_Register_2
Relative Address	0x0000007C
Absolute Address	ttc0: 0xF800107C ttc1: 0xF800207C
Width	16 bits
Access Type	ro
Reset Value	0x00000000
Description	pclk cycle count for event

Register Event_Register_2 Details

Field Name	Bits	Type	Reset Value	Description
Event	15:0	ro	0x0	This register stores the result of the pclk count during the ext_clk high or low pulse.

Register ([ttc](#)) Event_Register_3

Name	Event_Register_3
Relative Address	0x00000080
Absolute Address	ttc0: 0xF8001080 ttc1: 0xF8002080
Width	16 bits
Access Type	ro
Reset Value	0x00000000

Description pclk cycle count for event

Register Event_Register_3 Details

Field Name	Bits	Type	Reset Value	Description
Event	15:0	ro	0x0	This register stores the result of the pclk count during the ext_clk high or low pulse.

B.33 UART Controller (UART)

Module Name	UART Controller (UART)
Software Name	XUARTPS
Base Address	0xE0000000 uart0 0xE0001000 uart1
Description	Universal Asynchronous Receiver Transmitter Instance no. 0.
Version	1.0
Doc Version	1.2
Vendor Info	Cadence UART

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
Control_reg0	0x00000000	32	mixed	0x00000128	UART Control register
mode_reg0	0x00000004	32	mixed	0x00000000	UART Mode register
Intrpt_en_reg0	0x00000008	32	mixed	0x00000000	Interrupt Enable register
Intrpt_dis_reg0	0x0000000C	32	mixed	0x00000000	Interrupt disable register.
Intrpt_mask_reg0	0x00000010	32	ro	0x00000000	Interrupt mask register
Chnl_int_sts_reg0	0x00000014	32	wtc	0x00000200	Channel interrupt Status register
Baud_rate_gen_reg0	0x00000018	32	mixed	0x0000028B	Baud rate divider register.
Rcvr_timeout_reg0	0x0000001C	32	mixed	0x00000000	Receiver timeout register
Rcvr_FIFO_trigger_level0	0x00000020	32	mixed	0x00000020	Receiver FIFO trigger level register
Modem_ctrl_reg0	0x00000024	32	mixed	0x00000000	Modem control register
Modem_sts_reg0	0x00000028	32	mixed	x	Modem status register
Channel_sts_reg0	0x0000002C	32	ro	0x00000000	Channel Status register
TX_RX_FIFO0	0x00000030	32	mixed	0x00000000	Transmit and Receive FIFO
Baud_rate_divider_reg0	0x00000034	32	mixed	0x0000000F	baud rate divider register
Flow_delay_reg0	0x00000038	32	mixed	0x00000000	Flow Control Delay Register
Tx_FIFO_trigger_level0	0x00000044	32	mixed	0x00000020	Transmitter FIFO Trigger Level Register

Register ([UART](#)) Control_reg0

Name	Control_reg0
Software Name	CR
Relative Address	0x00000000
Absolute Address	uart0: 0xE0000000 uart1: 0xE0001000
Width	32 bits
Access Type	mixed
Reset Value	0x00000128
Description	UART Control register

Register Control_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:9	ro	0x0	Reserved, read as zero, ignored on write.
STPBRK (STOPBRK)	8	rw	0x1	Stop transmitter break: 0: start break transmission, 1: stop break transmission.
STTBRK (STARTBRK)	7	rw	0x0	Start transmitter break: 0: 1: start to transmit a break. Can only be set if STPBRK (Stop transmitter break) is not high.
RSTTO (TORST)	6	rw	0x0	Restart receiver timeout counter: 0: receiver timeout counter disabled, 1: receiver timeout counter is restarted.
TXDIS (TX_DIS)	5	rw	0x1	Transmit disable: 0: enable transmitter, 1: disable transmitter
TXEN (TX_EN)	4	rw	0x0	Transmit enable: 0: disable transmitter, 1: enable transmitter, provided the TXDIS field is set to 0.
RXDIS (RX_DIS)	3	rw	0x1	Receive disable: 0: enable, 1: disable
RXEN (RX_EN)	2	rw	0x0	Receive enable: 0: disable, 1: enable. When set to one, the receiver logic is enabled, provided the RXDIS field is set to zero.

Field Name	Bits	Type	Reset Value	Description
TXRES (TXRST)	1	rw	0x0	Software reset for Tx data path: 0: 1: transmitter logic is reset and all pending transmitter data is discarded self clear
RXRES (RXRST)	0	rw	0x0	Software reset for Rx data path: 0: 1: receiver logic is reset and all pending receiver data is discarded self clear

Register ([UART](#)) mode_reg0

Name	mode_reg0
Software Name	MR
Relative Address	0x00000004
Absolute Address	uart0: 0xE0000004 uart1: 0xE0001004
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	UART Mode register

Register mode_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:12	ro	0x0	Reserved, read as zero, ignored on write.
reserved	11	rw	0x0	Reserved. Do not modify.
reserved	10	rw	0x0	Reserved. Do not modify.
CHMODE	9:8	rw	0x0	Channel mode: 00: normal 01: automatic cho 10: local loopback 11: remote loopback
NBSTOP	7:6	rw	0x0	Number of stop bits: 00: 1 stop bit 01: 1.5 stop bits 10: 2 stop bits 11: reserved

Field Name	Bits	Type	Reset Value	Description
PAR	5:3	rw	0x0	Parity type select: 000: even parity 001: odd parity 010: forced to 0 parity (space) 011: forced to 1 parity (mark) 1xx: no parity
CHRL	2:1	rw	0x0	Character length select: 11: 6 bits 10: 7 bits 0x: 8 bits
CLKS (CLKSEL)	0	rw	0x0	Clock source select: 0: clock source is uart_clk 1: clock source is uart_clk/8

Register ([UART](#)) Intrpt_en_reg0

Name	Intrpt_en_reg0
Software Name	IER
Relative Address	0x00000008
Absolute Address	uart0: 0xE0000008 uart1: 0xE0001008
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Interrupt Enable register

Register Intrpt_en_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:13	ro	0x0	Reserved, read as zero, ignored on write.
TOVR	12	wo	0x0	Transmitter FIFO Overflow interrupt: 0: disable 1: enable
TNFUL	11	wo	0x0	Transmitter FIFO Nearly Full interrupt: 0: disable 1: enable
TTRIG	10	wo	0x0	Transmitter FIFO Trigger interrupt: 0: disable 1: enable
DMSI (IXR_DMS)	9	wo	0x0	Delta Modem Status Indicator interrupt: 0: disable 1: enable

Field Name	Bits	Type	Reset Value	Description
TIMEOUT (IXR_TOUT)	8	wo	0x0	Receiver Timeout Error interrupt: 0: disable 1: enable
PARE (IXR_PARITY)	7	wo	0x0	Receiver Parity Error interrupt: 0: disable 1: enable
FRAME (IXR_FRAMING)	6	wo	0x0	Receiver Framing Error interrupt: 0: disable 1: enable
ROVR (IXR_OVER)	5	wo	0x0	Receiver Overflow Error interrupt: 0: disable 1: enable
TFUL (IXR_TXFULL)	4	wo	0x0	Transmitter FIFO Full interrupt: 0: disable 1: enable
EMPTY (IXR_TXEMPTY)	3	wo	0x0	Transmitter FIFO Empty interrupt: 0: disable 1: enable
RFUL (IXR_RXFULL)	2	wo	0x0	Receiver FIFO Full interrupt: 0: disable, 1: enable
REMPY (IXR_RXEMPTY)	1	wo	0x0	Receiver FIFO Empty interrupt: 0: disable, 1: enable
RTRIG (IXR_RXOVR)	0	wo	0x0	Receiver FIFO Trigger interrupt: 0: disable, 1: enable

Register ([UART](#)) Intrpt_dis_reg0

Name	Intrpt_dis_reg0
Software Name	IDR
Relative Address	0x0000000C
Absolute Address	uart0: 0xE000000C uart1: 0xE000100C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Interrupt disable register.

Register Intrpt_dis_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:13	ro	0x0	Reserved, read as zero, ignored on write.
TOVR	12	wo	0x0	Transmitter FIFO Overflow interrupt: 0: enable; 1: disable

Field Name	Bits	Type	Reset Value	Description
TNFUL	11	wo	0x0	Transmitter FIFO Nearly Full interrupt: 0: enable, 1: disable
TTRIG	10	wo	0x0	Transmitter FIFO Trigger interrupt: 0: enable, 1: disable
DMSI (IXR_DMS)	9	wo	0x0	Delta Modem Status Indicator interrupt: 0: enable, 1: disable
TIMEOUT (IXR_TOUT)	8	wo	0x0	Receiver Timeout Error interrupt: 0: enable, 1: disable
PARE (IXR_PARITY)	7	wo	0x0	Receiver Parity Error interrupt: 0: enable, 1: disable
FRAME (IXR_FRAMING)	6	wo	0x0	Receiver Framing Error interrupt: 0: enable, 1: disable
ROVR (IXR_OVER)	5	wo	0x0	Receiver Overflow Error interrupt: 0: enable, 1: disable
TFUL (IXR_TXFULL)	4	wo	0x0	Transmitter FIFO Full interrupt: 0: enable, 1: disable
EMPTY (IXR_TXEMPTY)	3	wo	0x0	Transmitter FIFO Empty interrupt: 0: enable, 1: disable
RFUL (IXR_RXFULL)	2	wo	0x0	Receiver FIFO Full interrupt: 0: enable, 1: disable
REMPY (IXR_RXEMPTY)	1	wo	0x0	Receiver FIFO Empty interrupt: 0: enable, 1: disable
RTRIG (IXR_RXOVR)	0	wo	0x0	Receiver FIFO Trigger interrupt: 0: enable, 1: disable

Register ([UART](#)) Intrpt_mask_reg0

Name	Intrpt_mask_reg0
Software Name	IMR
Relative Address	0x00000010
Absolute Address	uart0: 0xE0000010 uart1: 0xE0001010
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Interrupt mask register

Register Intrpt_mask_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:13	ro	0x0	Reserved, read as zero, ignored on write.
TOVR	12	ro	0x0	Transmitter FIFO Overflow interrupt status: 0: interrupt is disabled 1: interrupt is enabled
TNFUL	11	ro	0x0	Transmitter FIFO Nearly Full interrupt mask status: 0: interrupt is disabled 1: interrupt is enabled
TTRIG	10	ro	0x0	Transmitter FIFO Trigger interrupt mask status: 0: interrupt is disabled 1: interrupt is enabled
DMSI (IXR_DMS)	9	ro	0x0	Delta Modem Status Indicator interrupt mask status: 0: interrupt is disabled 1: interrupt is enabled
TIMEOUT (IXR_TOUT)	8	ro	0x0	Receiver Timeout Error interrupt mask status: 0: interrupt is disabled 1: interrupt is enabled
PARE (IXR_PARITY)	7	ro	0x0	Receiver Parity Error interrupt mask status: 0: interrupt is disabled 1: interrupt is enabled
FRAME (IXR_FRAMING)	6	ro	0x0	Receiver Framing Error interrupt mask status: 0: interrupt is disabled 1: interrupt is enabled
ROVR (IXR_OVER)	5	ro	0x0	Receiver Overflow Error interrupt mask status: 0: interrupt is disabled 1: interrupt is enabled
TFUL (IXR_TXFULL)	4	ro	0x0	Transmitter FIFO Full interrupt mask status: 0: interrupt is disabled 1: interrupt is enabled
EMPTY (IXR_TXEMPTY)	3	ro	0x0	Transmitter FIFO Empty interrupt mask status: 0: interrupt is disabled 1: interrupt is enabled
RFUL (IXR_RXFULL)	2	ro	0x0	Receiver FIFO Full interrupt mask status: 0: interrupt is disabled 1: interrupt is enabled

Field Name	Bits	Type	Reset Value	Description
REMPY (IXR_RXEMPTY)	1	ro	0x0	Receiver FIFO Empty interrupt mask status: 0: interrupt is disabled 1: interrupt is enabled
RTRIG (IXR_RXOVR)	0	ro	0x0	Receiver FIFO Trigger interrupt mask status: 0: interrupt is enabled 1: interrupt is enabled

Register ([UART](#)) Chnl_int_sts_reg0

Name	Chnl_int_sts_reg0
Software Name	ISR
Relative Address	0x00000014
Absolute Address	uart0: 0xE0000014 uart1: 0xE0001014
Width	32 bits
Access Type	wtc
Reset Value	0x00000200
Description	Channel interrupt Status register

Register Chnl_int_sts_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:13	wtc	0x0	Reserved, read as zero, ignored on write.
TOVR	12	wtc	0x0	Transmitter FIFO Overflow interrupt mask status: 0: no interrupt occurred 1: interrupt occurred
TNFUL	11	wtc	0x0	Transmitter FIFO Nearly Full interrupt mask status: 0: no interrupt occurred 1: interrupt occurred
TTRIG	10	wtc	0x0	Transmitter FIFO Trigger interrupt mask status. 0: no interrupt occurred 1: interrupt occurred
DMSI (IXR_DMS)	9	wtc	0x1	Delta Modem Status Indicator interrupt mask status: 0: no interrupt occurred 1: interrupt occurred

Field Name	Bits	Type	Reset Value	Description
TIMEOUT (IXR_TOUT)	8	wtc	0x0	Receiver Timeout Error interrupt mask status: 0: no interrupt occurred 1: interrupt occurred
PARE (IXR_PARITY)	7	wtc	0x0	Receiver Parity Error interrupt mask status: 0: no interrupt occurred 1: interrupt occurred
FRAME (IXR_FRAMING)	6	wtc	0x0	Receiver Framing Error interrupt mask status: 0: no interrupt occurred 1: interrupt occurred
ROVR (IXR_OVER)	5	wtc	0x0	Receiver Overflow Error interrupt mask status: 0: no interrupt occurred 1: interrupt occurred
TFUL (IXR_TXFULL)	4	wtc	0x0	Transmitter FIFO Full interrupt mask status: 0: no interrupt occurred 1: interrupt occurred
EMPTY (IXR_TXEMPTY)	3	wtc	0x0	Transmitter FIFO Empty interrupt mask status: 0: no interrupt occurred 1: interrupt occurred
RFUL (IXR_RXFULL)	2	wtc	0x0	Receiver FIFO Full interrupt mask status: 0: no interrupt occurred 1: interrupt occurred
REEMPTY (IXR_RXEMPTY)	1	wtc	0x0	Receiver FIFO Empty interrupt mask status: 0: no interrupt occurred 1: interrupt occurred
RTRIG (IXR_RXOVR)	0	wtc	0x0	Receiver FIFO Trigger interrupt mask status: 0: no interrupt occurred 1: interrupt occurred

Register ([UART](#)) Baud_rate_gen_reg0

Name	Baud_rate_gen_reg0
Software Name	BAUDGEN
Relative Address	0x00000018
Absolute Address	uart0: 0xE0000018 uart1: 0xE0001018
Width	32 bits
Access Type	mixed

Reset Value 0x0000028B

Description Baud rate divider register.

Register Baud_rate_gen_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as zero, ignored on write.
CD	15:0	rw	0x28B	Baud Rate Clock Divisor Value: 0: Disables baud_sample 1: Clock divisor bypass 2 - 65535: baud_sample value

Register ([UART](#)) Rcvr_timeout_reg0

Name Rcvr_timeout_reg0

Software Name RXTOUT

Relative Address 0x0000001C

Absolute Address uart0: 0xE000001C
uart1: 0xE000101C

Width 32 bits

Access Type mixed

Reset Value 0x00000000

Description Receiver timeout register

Register Rcvr_timeout_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	ro	0x0	Reserved, read as zero, ignored on write.
RTO	7:0	rw	0x0	Receiver timeout value: 0: Disables receiver timeout counter 1 - 255: Receiver timeout

Register ([UART](#)) Rcvr_FIFO_trigger_level0

Name Rcvr_FIFO_trigger_level0

Software Name RXWM

Relative Address 0x00000020

Absolute Address uart0: 0xE0000020
uart1: 0xE0001020

Width 32 bits

Access Type mixed

Reset Value 0x00000020

Description Receiver FIFO trigger level register

Register Rcvr_FIFO_trigger_level0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:6	ro	0x0	Reserved, read as zero, ignored on write.
RTRIG	5:0	rw	0x20	Receiver FIFO trigger level value: 0: Disables receiver timeout counter 1 - 63: Receiver timeout

Register ([UART](#)) Modem_ctrl_reg0

Name Modem_ctrl_reg0

Software Name MODEMCR

Relative Address 0x00000024

Absolute Address uart0: 0xE0000024
 uart1: 0xE0001024

Width 32 bits

Access Type mixed

Reset Value 0x00000000

Description Modem control register

Register Modem_ctrl_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:6	ro	0x0	Reserved, read as zero, ignored on write.
FCM	5	rw	0x0	Automatic flow control mode: 0: disable 1: enable
reserved	4:2	ro	0x0	Reserved, read as zero, ignored on write.
RTS	1	rw	0x0	Request to send output control: 0: forced to logic 1 1: forced to logic 0
DTR	0	rw	0x0	Data Terminal Ready: 0: forced to logic 1 1: forced to logic 0

Register ([UART](#)) Modem_sts_reg0

Name	Modem_sts_reg0
Software Name	MODEMSR
Relative Address	0x00000028
Absolute Address	uart0: 0xE0000028 uart1: 0xE0001028
Width	32 bits
Access Type	mixed
Reset Value	x
Description	Modem status register

Register Modem_sts_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:9	ro	x	Reserved, read as zero, ignored on write.
FCMS	8	wtc	x	Flow Control Mode: 0: disabled 1: enabled
DCD	7	wtc	x	Data Carrier Detect Input status: 0: ua_ndcd input is high 1: ua_ndcd input is low
RI	6	wtc	x	Ring Indicator input status: 0: ua_nri input is high 1: ua_nri input is low
DSR	5	wtc	x	Data Set Ready input status: 0: ua_ndsr input is high 1: ua_ndsr input is low
CTS	4	wtc	x	Clear to Send input status: 0: ua_ncts input is high 1: ua_ncts input is low
DDCD (MEDEMSR_DCDX)	3	wtc	x	Delta Data Carrier Detect status: 0: No change has occurred 1: Change has occurred
TERI (MEDEMSR_RIX)	2	wtc	x	Trailing Edge Ring Indicator status: 0: No trailing edge has occurred 1: Trailing edge has occurred

Field Name	Bits	Type	Reset Value	Description
DDSR (MEDEMSR_DSRX)	1	wtc	x	Delta Data Set Ready status: 0: No change has occurred 1: Change has occurred
DCTS (MEDEMSR_CTSX)	0	wtc	x	Delta Clear To Send status: 0: No change has occurred 1: Change has occurred

Register (UART) Channel_sts_reg0

Name	Channel_sts_reg0
Software Name	SR
Relative Address	0x0000002C
Absolute Address	uart0: 0xE000002C uart1: 0xE000102C
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Channel Status register

Register Channel_sts_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:15	ro	0x0	Reserved, read as zero, ignored on write.
TNFUL	14	ro	0x0	Transmitter FIFO Nearly Full continuous status: 0: More than one byte is unused in the Tx FIFO 1: Only one byte is free in the Tx FIFO
TTRIG	13	ro	0x0	Transmitter FIFO Trigger continuous status: 0: Tx FIFO fill level is less than TTRIG 1: Tx FIFO fill level is greater than or equal to TTRIG
FDELT (FLOWDEL)	12	ro	0x0	Receiver flow delay trigger continuous status: 0: Rx FIFO fill level is less than FDEL 1: Rx FIFO fill level is greater than or equal to FDEL
TACTIVE	11	ro	0x0	Transmitter state machine active status: 0: inactive state 1: active state

Field Name	Bits	Type	Reset Value	Description
RACTIVE	10	ro	0x0	Receiver state machine active status: 0: inactive state 1: active state
DMSI (DMS)	9	ro	0x0	Delta Modem Status Indicator status: 0: no interrupt occurred 1: interrupt occurred
TIMEOUT (TOUT)	8	ro	0x0	Receiver Timeout status: 0: no interrupt occurred 1: interrupt occurred
PARE (PARITY)	7	ro	0x0	Receiver Parity Error status: 0: no interrupt occurred 1: interrupt occurred
FRAME	6	ro	0x0	Receiver Frame Error status: 0: no interrupt occurred 1: interrupt occurred
ROVR (OVER)	5	ro	0x0	Receiver Overflow Error status: 0: no interrupt occurred 1: interrupt occurred
TFUL (TXFULL)	4	ro	0x0	Transmitter FIFO Full continuous status: 0: Tx FIFO is not full 1: Tx FIFO is full
EMPTY (TXEMPTY)	3	ro	0x0	Transmitter FIFO Empty continuous status: 0: Tx FIFO is not empty 1: Tx FIFO is empty
RFUL (RXFULL)	2	ro	0x0	Receiver FIFO Full continuous status: 1: Rx FIFO is full 0: Rx FIFO is not full
REEMPTY (RXEMPTY)	1	ro	0x0	Receiver FIFO Full continuous status: 0: Rx FIFO is not empty 1: Rx FIFO is empty
RTRIG (RXOVR)	0	ro	0x0	Receiver FIFO Trigger continuous status: 0: Rx FIFO fill level is less than RTRIG 1: Rx FIFO fill level is greater than or equal to RTRIG

Register ([UART](#)) TX_RX_FIFO0

Name TX_RX_FIFO0

Software Name FIFO

Relative Address	0x00000030
Absolute Address	uart0: 0xE0000030 uart1: 0xE0001030
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Transmit and Receive FIFO

Register TX_RX_FIFO0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	ro	0x0	Reserved, read as zero, ignored on write.
FIFO	15:0	rw	0x0	Operates as Tx FIFO and Rx FIFO.

Register ([UART](#)) Baud_rate_divider_reg0

Name	Baud_rate_divider_reg0
Relative Address	0x00000034
Absolute Address	uart0: 0xE0000034 uart1: 0xE0001034
Width	32 bits
Access Type	mixed
Reset Value	0x0000000F
Description	baud rate divider register

Register Baud_rate_divider_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	ro	0x0	Reserved, read as zero, ignored on write.
BDIV	7:0	rw	0xF	Baud rate divider value: 0 - 3: ignored 4 - 255: Baud rate

Register ([UART](#)) Flow_delay_reg0

Name	Flow_delay_reg0
Relative Address	0x00000038
Absolute Address	uart0: 0xE0000038 uart1: 0xE0001038

Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	Flow Control Delay Register

Register Flow_delay_reg0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:6	ro	0x0	Reserved, read as zero, ignored on write.
FDEL	5:0	rw	0x0	Rx FIFO trigger level for ua_nrts de-assertion: 0 - 3: disabled 4 to 65535: ua_nrts is driven high when Rx FIFO fill level equals FDEL

Register ([UART](#)) Tx_FIFO_trigger_level0

Name	Tx_FIFO_trigger_level0
Relative Address	0x00000044
Absolute Address	uart0: 0xE0000044 uart1: 0xE0001044
Width	32 bits
Access Type	mixed
Reset Value	0x00000020
Description	Transmitter FIFO Trigger Level Register

Register Tx_FIFO_trigger_level0 Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:6	ro	0x0	Reserved, read as zero, ignored on write.
TTRIG	5:0	rw	0x20	Transmitter FIFO trigger level: 0: Disables transmitter FIFO trigger level function 1 - 63: Trigger set when transmitter FIFO fills to TTRIG bytes

B.34 USB Controller (usb)

Module Name	USB Controller (usb)
Software Name	XUSBPS
Base Address	0xE0002000 usb0 0xE0003000 usb1
Description	USB controller registers Instance no. 0.
Version	2.2
Doc Version	1.3
Vendor Info	Synopsys

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
ID	0x00000000	32	ro	0xE441FA05	The ID register identifies the USB-HS 2.0 core and its revision
HWGENERAL	0x00000004	12	ro	0x00000083	General hardware parameters as defined in configuration file.
HWHOST	0x00000008	32	ro	0x10020001	Host hardware parameters as defined in configuration file
HWDEVICE	0x0000000C	6	ro	0x00000019	Device hardware parameters as defined in configuration file.
HWTXBUF	0x00000010	32	ro	0x80060A10	TX buffer hardware parameters as defined in configuration file
HWRXBUF	0x00000014	32	ro	0x00000A10	RX buffer hardware parameters as defined in configuration file
GPTIMER0LD	0x00000080	24	rw	0x00000000	This register contains the timer duration or load value. See the GPTIMER0CTRL for a description of the timer functions
GPTIMER0CTRL	0x00000084	32	mixed	0x00000000	This register contains the control for the timer and a data field, which can be queried to determine the running count value.

Register Name	Address	Width	Type	Reset Value	Description
GPTIMER1LD	0x00000088	24	rw	0x00000000	This register contains the timer duration or load value. See the GPTIMER0CTRL for a description of the timer functions
GPTIMER1CTRL	0x0000008C	32	mixed	0x00000000	This register contains the control for the timer and a data field, which can be queried to determine the running count value.
SBUSCFG	0x00000090	3	rw	0x00000003	This register contains the control for the system bus interface (such as AMBA / BVCIMaster / Slave interfaces).
CAPLENGTH_HCVERSION	0x00000100	32	ro	0x01000040	Device/Host Capability registers specify the software limits, restrictions, and capabilities of the host/device controller implementation.
HCSPARAMS	0x00000104	28	ro	0x00010011	Port steering logic capabilities are described in this register.
HCCPARAMS	0x00000108	16	ro	0x00000006	This register identifies multiple mode control (time-base bit functionality) addressing capability
DCIVERSION	0x00000120	16	ro	0x00000001	The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.
DCCPARAMS	0x00000124	9	ro	0x0000018C	These fields describe the overall host/device capability of the controller
USBCMD	0x00000140	24	mixed	0x00000B00	The serial bus host/device controller executes the command indicated in this register
USBSTS	0x00000144	26	mixed	0x00000000	This register indicates various states of the Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus.

Register Name	Address	Width	Type	Reset Value	Description
USBINTR	0x00000148	26	mixed	0x00000000	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 Status register (USBSTS) still shows interrupt sources even if they are disabled by the USBINTR register, allowing polling of interrupt events by the software.
FRINDEX	0x0000014C	14	rw	0x00000000	This register is used by the host controller to index the periodic frame list. The register updates every 125 us (once each micro-frame).
PERIODICLISTBASE_DEVICEADDR	0x00000154	32	mixed	0x00000000	This register is has two different uses when working in device or host mode.
ASYNCLISTADDR_E_NDPOINTLISTADDR	0x00000158	32	mixed	0x00000000	This register is has two different uses when working in device or host mode.
TTCTRL	0x0000015C	32	mixed	0x00000000	This register contains parameters needed for internal TT operations.
BURSTSIZE	0x00000160	17	rw	0x00001010	This register controls the burst size used during data movement on the initiator/master interface.
TXFILLTUNING	0x00000164	22	mixed	0x00020000	The fields in this register control performance tuning associated with how the Controller posts data to the TX latency FIFO before moving the data onto the USB bus.
TXTTFILLTUNING	0x00000168	13	mixed	0x00000000	This register provides a function similar to TXFILLTUNING except there is no equivalent to TXFIFOTHRES because the TT TX latency FIFO is always loaded in a single burst. Even
IC_USB	0x0000016C	32	mixed	0x00000000	This register enable and controls the IC_USB FS/LS transceiver.

Register Name	Address	Width	Type	Reset Value	Description
ULPI_VIEWPORT	0x00000170	32	mixed	0x00000000	The register provides indirect access to the ULPI PHY register set. Although the core performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access.
ENDPTNAK	0x00000178	32	wtc	0x00000000	Only implemented in device mode.
ENDPTNAKEN	0x0000017C	32	rw	0x00000000	Only implemented in device mode.
CONFIGFLAG	0x00000180	32	ro	0x00000001	This register is not used in this implementation. A read from this register returns a constant of a 00000001h to indicate that all port routings default to this host controller.
PORTSC1	0x00000184	32	mixed	0x00000000	The Controller implement one The number of port registers implemented by a particular instantiation is documented in the HCSPARAM register. Software uses this information as an input parameter to determine how many ports need service. This implement contains only 1 host port.
OTGSC	0x000001A4	32	mixed	0x00000020	The Controller implements one On-The-Go (OTG) Status and Control register. The 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)
USBMODE	0x000001A8	32	mixed	0x00000000	USB Mode Selection register
ENDPTSETUPSTAT	0x000001AC	16	wtc	0x00000000	Only implemented in device mode.
ENDPTPRIME	0x000001B0	32	wtc	0x00000000	Only implemented in device mode

Register Name	Address	Width	Type	Reset Value	Description
ENDPTFLUSH	0x000001B4	32	wtc	0x00000000	The Flush operation for an endpoint will clear the Ready status of that endpoint and re-align the Latency Buffer pointers, but not clear the actual data that resides in the Latency Buffers.
ENDPTSTAT	0x000001B8	32	ro	0x00000000	Only implemented in device mode.
ENDPTCOMPLETE	0x000001BC	32	ro	0x00000000	Only implemented in device mode.
ENDPTCTRL0	0x000001C0	24	mixed	0x00800080	Every device will implement Endpoint0 as a control endpoint. Only implemented in device mode.
ENDPTCTRL1	0x000001C4	24	mixed	0x00000000	This is an ENDPTCTRL1 register for the endpoint 1 device. Only implemented in device mode.
ENDPTCTRL2	0x000001C8	24	mixed	0x00000000	This is an ENDPTCTRL2 register for the endpoint 2 device. Only implemented in device mode.
ENDPTCTRL3	0x000001CC	24	mixed	0x00000000	This is an ENDPTCTRL3 register for the endpoint 3 device. Only implemented in device mode.
ENDPTCTRL4	0x000001D0	24	mixed	0x00000000	This is an ENDPTCTRL4 register for the endpoint 4 device. Only implemented in device mode.
ENDPTCTRL5	0x000001D4	24	mixed	0x00000000	This is an ENDPTCTRL5 register for the endpoint 5 device. Only implemented in device mode.
ENDPTCTRL6	0x000001D8	24	mixed	0x00000000	This is an ENDPTCTRL6 register for the endpoint 6 device. Only implemented in device mode.
ENDPTCTRL7	0x000001DC	24	mixed	0x00000000	This is an ENDPTCTRL7 register for the endpoint 7 device. Only implemented in device mode.
ENDPTCTRL8	0x000001E0	24	mixed	0x00000000	This is an ENDPTCTRL8 register for the endpoint 8 device. Only implemented in device mode.

Register Name	Address	Width	Type	Reset Value	Description
ENDPTCTRL9	0x000001E4	24	mixed	0x00000000	This is an ENDPTCTRL9 register for the endpoint 9 device. Only implemented in device mode.
ENDPTCTRL10	0x000001E8	24	mixed	0x00000000	This is an ENDPTCTRL10 register for the endpoint 10 device. Only implemented in device mode.
ENDPTCTRL11	0x000001EC	24	mixed	0x00000000	This is an ENDPTCTRL11 register for the endpoint 11 device. Only implemented in device mode.
ENDPTCTRL12	0x000001F0	24	mixed	0x00000000	This is an ENDPTCTRL12 register for the endpoint 12 device. Only implemented in device mode.

Register ([usb](#)) ID

Name	ID
Relative Address	0x00000000
Absolute Address	usb0: 0xE0002000 usb1: 0xE0003000
Width	32 bits
Access Type	ro
Reset Value	0xE441FA05
Description	The ID register identifies the USB-HS 2.0 core and its revision

Register ID Details

Field Name	Bits	Type	Reset Value	Description
CIVERSION	31:29	ro	0x7	Identifies the CI version
VERSION	28:25	ro	0x2	Identifies the version of the core
REVISION	24:21	ro	0x2	Revision number of the core
TAG	20:16	ro	0x1	Identifies the tag of the core ; Current 2.20a version
reserved	15:14	ro	0x3	RESERVED. Writes are ignored
NID	13:8	ro	0x3A	NID Complement of ID; Ones complement version of ID

Field Name	Bits	Type	Reset Value	Description
reserved	7:6	ro	0x0	RESERVED. Writes are ignored, read data is always zero.
ID	5:0	ro	0x5	ID Configuration number ; This number indicates the type of the USB-HS 2.0 core.

Register ([usb](#)) HWGENERAL

Name	HWGENERAL
Relative Address	0x00000004
Absolute Address	usb0: 0xE0002004 usb1: 0xE0003004
Width	12 bits
Access Type	ro
Reset Value	0x00000083
Description	General hardware parameters as defined in configuration file.

Register HWGENERAL Details

Field Name	Bits	Type	Reset Value	Description
SM	11:10	ro	0x0	VUSB_HS_PHY_SERIAL - This constant selects that the serial engine is used or not. 0 - No Serial Engine- Always use parallel signaling
PHYM	9:6	ro	0x2	VUSB_HS_PHY_TYPE - PHY interface type. 2 = ULPI
PHYW	5:4	ro	0x0	VUSB_HS_PHY16_8 - This constant selects which phy is being used. 0 = 8 bit wide data bus [60MHz clock from the transceiver]
BWT	3	ro	0x0	RESERVED.
CLKC	2:1	ro	0x1	VUSB_HS_CLOCK_CONFIGURATION - constant determines the clocking used in the core. 1 = xcvr_clk_0 < pe_clk = clk
RT	0	ro	0x1	VUSB_HS_RESET_TYPE - Reset Type. 1 = Use Asynchronous Resets

Register ([usb](#)) HWHOST

Name	HWHOST
------	--------

Relative Address	0x00000008
Absolute Address	usb0: 0xE0002008 usb1: 0xE0003008
Width	32 bits
Access Type	ro
Reset Value	0x10020001
Description	Host hardware parameters as defined in configuration file

Register HWHOST Details

Field Name	Bits	Type	Reset Value	Description
TPPER	31:24	ro	0x10	VUSB_HS_TT_PERIODIC_CONTEXTS
TTASY	23:16	ro	0x2	VUSB_HS_TT_ASYNC_CONTEXTS
	15:4	ro	0x0	RESERVED
NPORT	3:1	ro	0x0	VUSB_HS_NUM_PORT-1. 1- The VUSB_HS_NUM_PORT constant specifies the number of downstream ports by host port
HC	0	ro	0x1	VUSB_HS_HOST

Register ([usb](#)) HWDEVICE

Name	HWDEVICE
Relative Address	0x0000000C
Absolute Address	usb0: 0xE000200C usb1: 0xE000300C
Width	6 bits
Access Type	ro
Reset Value	0x00000019
Description	Device hardware parameters as defined in configuration file.

Register HWDEVICE Details

Field Name	Bits	Type	Reset Value	Description
DEVEP	5:1	ro	0xC	VUSB_HS_DEV_EP - Number of endpoints. 12 = VUSB_HS_DEV_EP constant represents the 12 number of endpoints
DC	0	ro	0x1	Device capable

Register ([usb](#)) HWTXBUF

Name	HWTXBUF
Relative Address	0x00000010
Absolute Address	usb0: 0xE0002010 usb1: 0xE0003010
Width	32 bits
Access Type	ro
Reset Value	0x80060A10
Description	TX buffer hardware parameters as defined in configuration file

Register HWTXBUF Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	ro	0x1	RESERVED
reserved	30:24	ro	0x0	RESERVED
TXCHANADD	23:16	ro	0x6	VUSB_HS_TX_CHAN_ADD - Address bits for each Endpoint; 64 words buffer for each end point. 6 - To address 64 depth TX buffer for each end point
TXADD	15:8	ro	0xA	VUSB_HS_TX_ADD ; 768 Depth TX buffer. 10- Address of the depth of TX buffer
TXBURST	7:0	ro	0x10	VUSB_HS_TX_BURST. Burst size for Memory To TX Buffer Transfers

Register ([usb](#)) HWRXBUF

Name	HWRXBUF
Relative Address	0x00000014
Absolute Address	usb0: 0xE0002014 usb1: 0xE0003014
Width	32 bits
Access Type	ro
Reset Value	0x00000A10
Description	RX buffer hardware parameters as defined in configuration file

Register HWRXBUF Details

Field Name	Bits	Type	Reset Value	Description
	31:24	ro	0x0	RESERVED
RXADD	15:8	ro	0xA	VUSB_HS_RX_ADD ; 768 Depth TX buffer. 10- Address of the depth of TX buffer
RXBURST	7:0	ro	0x10	VUSB_HS_RX_BURST. 16 = Burst size of 16 for Memory To TX Buffer Transfers

Register ([usb](#)) GPTIMER0LD

Name	GPTIMER0LD
Relative Address	0x00000080
Absolute Address	usb0: 0xE0002080 usb1: 0xE0003080
Width	24 bits
Access Type	rw
Reset Value	0x00000000
Description	This register contains the timer duration or load value. See the GPTIMER0CTRL for a description of the timer functions

Register GPTIMER0LD Details

Field Name	Bits	Type	Reset Value	Description
GPTLD	23:0	rw	0x0	General Purpose Timer Load Value register. This field is the value to be loaded into the GPTCNT countdown timer on a reset action. The value in this register represents the time in microseconds minus 1 for the timer duration. Example: for a one millisecond timer, load 1000-1=999 or 0x0003E7. Note: Max value is 0xFFFFF or 16.777215 seconds

Register ([usb](#)) GPTIMER0CTRL

Name	GPTIMER0CTRL
Relative Address	0x00000084
Absolute Address	usb0: 0xE0002084 usb1: 0xE0003084

Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	This register contains the control for the timer and a data field, which can be queried to determine the running count value.

Register GPTIMER0CTRL Details

Field Name	Bits	Type	Reset Value	Description
GPTRUN	31	rw	0x0	General Purpose Timer Run. This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT counter value.
GPTRST	30	wo	0x0	General Purpose Timer Reset. Writing a one to this bit will reload the GPTCNT with the value in GPTLD.
	29:25	ro	0x0	reserved
GPTMODE	24	rw	0x0	0b' - One Shot; '1b' - Repeat. This bit selects between a single timer countdown and a looped countdown. In one-shot mode, the timer will count down to zero, generate an interrupt and stop until the counter is reset by software. In repeat mode, the timer will count down to zero, generate an interrupt and automatically reload the counter to begin again.
GPTCNT	23:0	rw	0x0	General Purpose Timer Counter. This field is the value of the running timer.

Register ([usb](#)) GPTIMER1LD

Name	GPTIMER1LD
Relative Address	0x00000088
Absolute Address	usb0: 0xE0002088 usb1: 0xE0003088
Width	24 bits
Access Type	rw
Reset Value	0x00000000
Description	This register contains the timer duration or load value. See the GPTIMER0CTRL for a description of the timer functions

Register GPTIMER1LD Details

Field Name	Bits	Type	Reset Value	Description
GPTLD	23:0	rw	0x0	General Purpose Timer Load Value register. This field is the value to be loaded into the GPTCNT countdown timer on a reset action. The value in this register represents the time in microseconds minus 1 for the timer duration. Example: for a one millisecond timer, load 1000-1=999 or 0x0003E7. Note: Max value is 0xFFFFF or 16.777215 seconds

Register ([usb](#)) GPTIMER1CTRL

Name	GPTIMER1CTRL
Relative Address	0x0000008C
Absolute Address	usb0: 0xE000208C usb1: 0xE000308C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	This register contains the control for the timer and a data field, which can be queried to determine the running count value.

Register GPTIMER1CTRL Details

Field Name	Bits	Type	Reset Value	Description
GPTRUN	31	rw	0x0	General Purpose Timer Run. This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT counter value.
GPTRST	30	wo	0x0	General Purpose Timer Reset. Writing a one to this bit will reload the GPTCNT with the value in GPTLD.
	29:25	ro	0x0	reserved

Field Name	Bits	Type	Reset Value	Description
GPTMODE	24	rw	0x0	0b' - One Shot; '1b' - Repeat. This bit selects between a single timer countdown and a looped countdown. In one-shot mode, the timer will count down to zero, generate an interrupt and stop until the counter is reset by software. In repeat mode, the timer will count down to zero, generate an interrupt and automatically reload the counter to begin again.
GPTCNT	23:0	rw	0x0	General Purpose Timer Counter. This field is the value of the running timer.

Register ([usb](#)) SBUSCFG

Name	SBUSCFG
Relative Address	0x00000090
Absolute Address	usb0: 0xE0002090 usb1: 0xE0003090
Width	3 bits
Access Type	rw
Reset Value	0x00000003
Description	This register contains the control for the system bus interface (such as AMBA / BVIC Master / Slave interfaces).

Register SBUSCFG Details

Field Name	Bits	Type	Reset Value	Description
AHBBRST	2:0	rw	0x3	VUSB_HS_RX_BURST 0: incremental bursts of unspecified length 1: INCR4, non multiple transfers of 4 are decomposed to singles 2: INCR8, non multiple transfers of 8 are decomposed to INCR4 or singles 3: INCR16, non multiple transfers of 16 are decomposed to INCR8, INCR4 or singles 4: Do not use this value! 5: INCR4, non multiple transfers of 4 are decomposed to INCR of unspecified length 6: INCR8, non multiple transfers of 8 are decomposed to INCR4 or INCR of unspecified length 7: INCR16, non multiple transfers of 16 are decomposed to INCR8, INCR4 or INCR of unspecified length

Register ([usb](#)) CAPLENGTH_HCIVERSION

Name	CAPLENGTH_HCIVERSION
Relative Address	0x00000100
Absolute Address	usb0: 0xE0002100 usb1: 0xE0003100
Width	32 bits
Access Type	ro
Reset Value	0x01000040
Description	Device/Host Capability registers specify the software limits, restrictions, and capabilities of the host/device controller implementation.

Register CAPLENGTH_HCIVERSION Details

Field Name	Bits	Type	Reset Value	Description
HCIVERSION	31:16	ro	0x100	This is a two-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.
CAPLENGTH	15:0	ro	0x40	This register indicates capability register length. This register is used to indicate which offset to add to the register base address at the beginning of the Operational Register

Register ([usb](#)) HCSPARAMS

Name	HCSPARAMS
Relative Address	0x00000104
Absolute Address	usb0: 0xE0002104 usb1: 0xE0003104
Width	28 bits
Access Type	ro
Reset Value	0x00010011
Description	Port steering logic capabilities are described in this register.

Register HCSPARAMS Details

Field Name	Bits	Type	Reset Value	Description
N_TT	27:24	ro	0x0	This field indicates the number of embedded transaction translators associated with the USB2.0 host controller. For Multi-Port Host this field will always be equal to '0001b'. For all other implementation, N_TT = '0000b'. This field will always be '0'
N_PTT	23:20	ro	0x0	Number of Ports per Transaction Translator. This field indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. For Multi-Port Host this field will always equal N_PORTS. For all other implementations, N_PTT = '0000b'. This in a non-EHCI field to support embedded TT.
reserved	19:17	ro	0x0	RESERVED
PI	16	ro	0x1	Port indicator. This bit indicates whether the ports support port indicator control. When set to one, the port status and control registers include a read/writable field for controlling the state of the port indicator. This field will always be '1'
N_CC	15:12	ro	0x0	This field indicates the number of companion controllers associated with this USB2.0 host controller. A zero in this field indicates there are no internal Companion Controllers. Port ownership hand-off is not supported. A value larger than zero in this field indicates there are companion USB1.1 host controller(s). Port-ownership hand-offs are supported. High, Full and Low speed devices are supported on the host controller root ports.
N_PCC	11:8	ro	0x0	This field indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software.
reserved	7:5	ro	0x0	RESERVED
PPC	4	ro	0x1	Port Power Control. This field indicates whether the host controller implementation includes port power control. 1 = indicates the ports have port power switches. 0 = indicates the ports do not have port power switches.
N_PORTS	3:0	ro	0x1	Number of downstream ports supported by the host controller

Register ([usb](#)) HCCPARAMS

Name	HCCPARAMS
Relative Address	0x00000108
Absolute Address	usb0: 0xE0002108 usb1: 0xE0003108
Width	16 bits
Access Type	ro
Reset Value	0x00000006
Description	This register identifies multiple mode control (time-base bit functionality) addressing capability

Register HCCPARAMS Details

Field Name	Bits	Type	Reset Value	Description
EECP	15:8	ro	0x0	No Description
IST	7:4	ro	0x0	<p>Isochronous Scheduling Threshold.</p> <p>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 zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field will always be 4'b0</p>
	3	ro	0x0	reserved
ASP	2	ro	0x1	<p>Asynchronous Schedule Park Capability. If this bit is set to a one, then 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 USBCMD register.</p> <p>This field will always be '1'</p>

Field Name	Bits	Type	Reset Value	Description
PFL	1	ro	0x1	<p>If this bit is set to zero, then the system software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero.</p> <p>If set to a one, then the system software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This field will always be '1'</p>
ADC	0	ro	0x0	64-bit Addressing Capability. For this controller its always 0

Register ([usb](#)) DCIVERSION

Name	DCIVERSION
Relative Address	0x00000120
Absolute Address	usb0: 0xE0002120 usb1: 0xE0003120
Width	16 bits
Access Type	ro
Reset Value	0x00000001
Description	The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

Register DCIVERSION Details

Field Name	Bits	Type	Reset Value	Description
DCIVERSION	15:0	ro	0x1	The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register

Register ([usb](#)) DCCPARAMS

Name	DCCPARAMS
Relative Address	0x00000124
Absolute Address	usb0: 0xE0002124 usb1: 0xE0003124
Width	9 bits

Access Type	ro
Reset Value	0x0000018C
Description	These fields describe the overall host/device capability of the controller

Register DCCPARAMS Details

Field Name	Bits	Type	Reset Value	Description
HC	8	ro	0x1	Host Capable. When this bit is 1, this controller is capable of operating as an EHCI compatible USB 2.0 host controller 1= The controller is USB 2.0 host controller capable
DC	7	ro	0x1	Device Capable register. When this bit is 1, this controller is capable of operating as a USB 2.0 device. 1= The controller is USB 2.0 device capable
	6:5	ro	0x0	reserved
DEN	4:0	ro	0xC	This field indicates the number of endpoints built into the device controller. 12 = 12 endpoints are supported

Register ([usb](#)) USBCMD

Name	USBCMD
Software Name	CMD
Relative Address	0x00000140
Absolute Address	usb0: 0xE0002140 usb1: 0xE0003140
Width	24 bits
Access Type	mixed
Reset Value	0x00000B00
Description	The serial bus host/device controller executes the command indicated in this register

Register USBCMD Details

Field Name	Bits	Type	Reset Value	Description
ITC	23:16	rw	0x0	Interrupt Threshold Control. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. 00h Immediate (no threshold) 01h 1 micro-frame 02h 2 micro-frames 04h 4 micro-frames 08h 8 micro-frames 10h 16 micro-frames 20h 32 micro-frames 40h 64 micro-frames
FS2	15	rw	0x0	MSB bit of FS field.
ATDTW	14	rw	0x0	This bit is used as a semaphore to ensure the proper addition of a new dTD to an active (primed) endpoint's linked list.
SUTW	13	rw	0x0	Setup TripWire. This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (See USBMODE register) then there exists a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists. Used only in device Mode
reserved	12	ro	0x0	RESERVED
ASPE	11	rw	0x1	If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this bit defaults to a 1b and is R/W. Otherwise the bit must be a zero and is RO. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. This field is set to '1'b in this implementation. Only used in host mode.
reserved	10	ro	0x0	RESERVED

Field Name	Bits	Type	Reset Value	Description
ASP	9:8	rw	0x3	<p>If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is RO. 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.</p> <p>This field is set to 3h in this implementation.</p> <p>Only used in host mode.</p>
LR	7	ro	0x0	Light Host/Device Controller Reset. This field will always be '0b'.
IAA	6	rw	0x0	<p>This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances the asynchronous schedule. Software must write a '1b' to this bit to ring the doorbell.</p> <p>This is used only in Host mode</p>
ASE	5	rw	0x0	<p>This bit controls whether the host controller skips processing the Asynchronous Schedule.</p> <p>Values Meaning :</p> <p>0- Do not process the Asynchronous Schedule.</p> <p>1- Use the ASYNCLISTADDR register to access the Asynchronous Schedule.</p> <p>Only used in host mode.</p>
PSE	4	rw	0x0	<p>This bit controls whether the host controller skips processing the Periodic Schedule. Values Meaning</p> <p>0 -> Do not process the Periodic Schedule</p> <p>1 -> Use the PERIODICLISTBASE register to access the Periodic Schedule.</p> <p>Only used in host mode.</p>
FS0 (FS01)	3:2	rw	0x0	<p>reserved</p> <p>0 -> 1024 elements (4096 bytes)</p> <p>1 -> 512 elements (2048 bytes)</p> <p>2 -> 256 elements (1024 bytes)</p> <p>3 -> 128 elements (512 bytes)</p> <p>4 -> 64 elements (256 bytes)</p> <p>5 -> 32 elements (128 bytes)</p> <p>6 -> 16 elements (64 bytes)</p> <p>7 -> 8 elements (32 bytes)</p>

Field Name	Bits	Type	Reset Value	Description
RST	1	rw	0x0	Software uses this bit to reset the controller. This bit is set to zero by the Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register.
RS	0	rw	0x0	Run/Stop bit , When set to a 1, the Controller proceeds with the execution of the schedule.

Register ([usb](#)) USBSTS

Name	USBSTS
Software Name	ISR
Relative Address	0x00000144
Absolute Address	usb0: 0xE0002144 usb1: 0xE0003144
Width	26 bits
Access Type	mixed
Reset Value	0x00000000
Description	This register indicates various states of the Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus.

Register USBSTS Details

Field Name	Bits	Type	Reset Value	Description
TI1 (IXR_TI1)	25	rw	0x0	This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
TI0 (IXR_TI0)	24	rw	0x0	This bit is set when the counter in the GPTIMER0CTRL register transitions to zero. Writing a one to this bit will clear it.
reserved	23:20	ro	0x0	T
UPI (IXR_UP)	19	rw	0x0	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 expected. This bit is not used by the device controller and will always be zero.

Field Name	Bits	Type	Reset Value	Description
UAI (IXR_UA)	18	rw	0x0	<p>USB Host Asynchronous Interrupt . 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.</p> <p>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 expected. This bit is not used by the device controller and will always be zero.</p>
reserved	17	ro	0x0	RESERVED
NAKI (IXR_NAK)	16	ro	0x0	<p>NAK Interrupt. 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. This bit is not used by the host controller and will always be zero.</p>
AS (IXR_AS)	15	ro	0x0	<p>This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled.</p> <p>The Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). Only used in host mode.</p>
PS (IXR_PS)	14	ro	0x0	<p>Periodic Schedule Status. This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). Only used in host mode.</p>
RCL (IXR_RCL)	13	ro	0x0	<p>Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. Used in Host mode only</p>

Field Name	Bits	Type	Reset Value	Description
HCH (IXR_HCH)	12	ro	0x0	HCHalted. This bit is a zero whenever the Run/Stop bit is a one. The Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Controller hardware (e.g. internal error). Only used in host mode.
reserved	11	ro	0x0	RESERVED
ULPII (IXR_ULPI)	10	rw	0x0	ULPI Interrupt. When the ULPI Viewport is present in the design, an event completion will set this interrupt.
reserved	9	ro	0x0	RESERVED
SLI (IXR_SLE)	8	rw	0x0	DCSuspend. When a device controller enters a suspend state from an active state, this bit will be set to a one. This bit is only cleared by software writing a 1 to it. Only used in device mode.
SRI (IXR_SR)	7	rw	0x0	SOF Received. When the device controller detects a Start Of (u)Frame, this bit will be set to a one. When a SOF is extremely late, the Controller, when in device mode, will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in FS mode and every 125us in HS mode and will be synchronized to the actual SOF that is received. Since the Controller when in device mode is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp. In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it.
URI (IXR_UR)	6	rw	0x0	USB Reset Received. When the Controller detects a USB Reset and enters the default state, this bit will be set to a one. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used in device mode.
AAI (IXR_AA)	5	rw	0x0	Interrupt on Async Advance. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used in host mode.

Field Name	Bits	Type	Reset Value	Description
SEI	4	rw	0x0	System Error. In the BVCI implementation of the USBHS core, this bit is not used, and will always be cleared to '0b'. In the AMBA implementation, this bit will be set to '1b' when an Error response is seen by the master interface
FRI (IXR_FRE)	3	rw	0x0	Frame List Rollover The Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. 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 USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX[13] toggles. Similarly, if the size is 512, the Controller sets this bit to a one every time FRINDEX[12] toggles. Only used in host mode.
PCI (IXR_PC)	2	rw	0x0	Port Change Detect. The Controller in host mode sets this bit to a one 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. The Controller in device mode sets this bit to a one when it detects resume signaling or the port controller enters the full or high-speed operational state. When the port controller exits the 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.
UEI (IXR_UE)	1	rw	0x0	USB Error Interrupt. When completion of a USB transaction results in an error condition, this bit is set by the Controller
UI (IXR_UI)	0	rw	0x0	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 Host Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than expected.

Register ([usb](#)) USBINTR

Name	USBINTR
Software Name	IER
Relative Address	0x00000148

Absolute Address	usb0: 0xE0002148 usb1: 0xE0003148
Width	26 bits
Access Type	mixed
Reset Value	0x00000000
Description	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 Status register (USBSTS) still shows interrupt sources even if they are disabled by the USBINTR register, allowing polling of interrupt events by the software.

Register USBINTR Details

Field Name	Bits	Type	Reset Value	Description
TIE1 (IXR_TI1)	25	rw	0x0	General Purpose Timer Interrupt Enable 1 When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
TIE0 (IXR_TI0)	24	rw	0x0	General Purpose Timer Interrupt Enable 0 When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI0 bit.
reserved	23:20	ro	0x0	Reserved
UPEI (IXR_UP)	19	rw	0x0	USB Host Periodic Interrupt Enable. When this bit is a one, and the UPI bit in the EXTSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the UPI bit.
UAEI (IXR_UA)	18	rw	0x0	USB Host Asynchronous Interrupt Enable When this bit is a one, and the UAI bit in the EXTSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the UAI bit.
reserved	17	ro	0x0	RESERVED
NAKEI (IXR_NAK)	16	ro	0x0	NAK Interrupt Enable when this bit is a one, and the NAKI bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the NAKI bit.
reserved	11	ro	0x0	RESERVED

Field Name	Bits	Type	Reset Value	Description
ULPIE (IXR_ULPI)	10	rw	0x0	ULPI Interrupt enable. When this bit is a one, and the ULPI Interrupt bit in the USBSTS register transitions, the Controller will issue an interrupt. The interrupt is acknowledged by software writing a one to the ULPI Interrupt bit.
reserved	9	ro	0x0	RESERVED
SLE (IXR_SLE)	8	rw	0x0	DCSuspend Interrupt Enable. When this bit is a one, and the SLI bit in the USBSTS register transitions, the Controller will issue an interrupt. The interrupt is acknowledged by software writing a one to the SLI bit. Only used in device mode.
SRE (IXR_SR)	7	rw	0x0	USB Reset Received Interrupt Enable When this bit is a one, and the SRI bit in the USBSTS register is a one, the Controller will issue an interrupt. The interrupt is acknowledged by software clearing the SRI bit.
URE (IXR_UR)	6	rw	0x0	USB Reset Received. When the Controller detects a USB Reset and enters the default state, this bit will be set to a one. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used in device mode.
AAE (IXR_AA)	5	rw	0x0	Interrupt on Async Advance Enable. When this bit is a one, and the AAI bit in the USBSTS register is a one, the Controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on AAI bit. Only used in host mode.
SEE	4	rw	0x0	System Error Interrupt Enable. When this bit is a one, and the SEI bit in the USBSTS register is a one, the Controller will issue an interrupt. The interrupt is acknowledged by software clearing the SSI bit.
FRE (IXR_FRE)	3	rw	0x0	Frame List Rollover Interrupt Enable When this bit is a one, and the FRI bit in the USBSTS register is a one, the Controller will issue an interrupt. The interrupt is acknowledged by software clearing the FRI bit. Only used in host mode.
PCE (IXR_PC)	2	rw	0x0	Port Change Detect Interrupt Enable When this bit is a one, and the PCI bit in the USBSTS register is a one, the Controller will issue an interrupt. The interrupt is acknowledged by software clearing the PCI bit.

Field Name	Bits	Type	Reset Value	Description
UEE (IXR_UE)	1	wtc	0x0	USB Error Interrupt When this bit is a one, and the UEI bit in the USBSTS register is a one, the Controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the UEI bit in the USBSTS register.
UE (IXR_UI)	0	rw	0x0	When this bit is a one, and the UI bit in the USBSTS register is a one, the Controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the UI bit.

Register ([usb](#)) FRINDEX

Name	FRINDEX
Software Name	FRAME
Relative Address	0x0000014C
Absolute Address	usb0: 0xE000214C usb1: 0xE000314C
Width	14 bits
Access Type	rw
Reset Value	0x00000000
Description	This register is used by the host controller to index the periodic frame list. The register updates every 125 us (once each micro-frame).

Register FRINDEX Details

Field Name	Bits	Type	Reset Value	Description
FRINDEX	13:0	rw	0x0	<p>The value, in this register, increments at the end of each time frame (e.g. micro-frame). 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 micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode.</p> <p>usbcmd-> Frame List-> Size</p> <p>000b ->(1024) -> 12</p> <p>001b ->(512) -> 11</p> <p>010b ->(256) -> 10</p> <p>011b ->(128) -> 9</p> <p>100b ->(64) -> 8</p> <p>101b ->(32) -> 7</p> <p>110b ->(16) -> 6</p> <p>111b ->(8) -> 5</p> <p>In device mode the value is the current frame number of the last frame transmitted. It is not used as an index.</p> <p>This register is read-only in device mode.</p>

Register ([usb](#)) PERIODICLISTBASE_DEVICEADDR

Name	PERIODICLISTBASE_DEVICEADDR
Software Name	LISTBASE
Relative Address	0x00000154
Absolute Address	usb0: 0xE0002154 usb1: 0xE0003154
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	This register is has two different uses when working in device or host mode.

Register PERIODICLISTBASE_DEVICEADDR Details

Field Name	Bits	Type	Reset Value	Description
PERBASE_USBADRA	31:25	rw	0x0	<p>In host mode: Periodic List Base Address: These bits correspond to memory address signals [31:12], respectively.</p> <p>In Device Mode: Device address advance</p> <p>When this bit is '0b', 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 will be loaded from the hidden register.</p> <p>Hardware will automatically clear this bit on the following conditions:</p> <ol style="list-style-type: none"> 1) IN is ACKed to endpoint 0. (USBADR is updated from hidden register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0).
PERBASE_USBADR	24	rw	0x0	<p>In host mode: Periodic List Base Address: These bits correspond to memory address signals [31:12], respectively.</p> <p>In device Mode: USB Device address These bits correspond to the USB device address</p>
PERBASE_Reserved	23:12	rw	0x0	<p>In host mode: Periodic List Base Address: These bits correspond to memory address signals [31:12], respectively.</p> <p>In Device Mode: Reserved</p>
reserved	11:0	ro	0x0	RESERVED

Register ([usb](#)) ASYNCLISTADDR_ENDPOINTLISTADDR

Name	ASYNCLISTADDR_ENDPOINTLISTADDR
Software Name	ASYNCLISTADDR
Relative Address	0x00000158
Absolute Address	usb0: 0xE0002158 usb1: 0xE0003158
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	This register is has two different uses when working in device or host mode.

Register ASYNCLISTADDR_ENDPOINTLISTADDR Details

Field Name	Bits	Type	Reset Value	Description
ASYBASE_EPBASE	31:11	rw	0x0	In host mode: Asynchronous List Base Address: These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH). In Device Mode: Endpoint List Base Address These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). (i.e. one queue head per endpoint & direction).
ASYBASE	10:5	rw	0x0	In host mode: Asynchronous List Base Address: These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH). In Device Mode: Reserved
reserved	4:0	ro	0x0	RESERVED

Register ([usb](#)) TTCTRL

Name	TTCTRL
Relative Address	0x0000015C
Absolute Address	usb0: 0xE000215C usb1: 0xE000315C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	This register contains parameters needed for internal TT operations.

Register TTCTRL Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	ro	0x0	RESERVED
TTHA (HUBADDR)	30:24	rw	0x0	Internal TT Hub Address Representation. This field is used to match against the Hub Address field in QH & 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 then the packet will be broadcast on the High Speed ports destined for a downstream High Speed hub with the address in the QH/siTD.
reserved	23:2	ro	0x0	RESERVED
TTAS	1	rw	0x0	Embedded TT Asynchronous Buffers Clear. This field will clear all pending transactions in the embedded TT Asynchronous Buffer(s). The clear will take as much time as necessary to clear buffer without interfering with a transaction in progress. TTAC will return to zero after being set by software only after the actual clear occurs.
TTAC	0	ro	0x0	Embedded TT Async Buffers Status. This read only bit will be '1' if one or more transactions are being held in the embedded TT Asynchronous Buffers. When this bit is a zero, then all outstanding transactions in the embedded TT have been flushed.

Register ([usb](#)) BURSTSIZE

Name	BURSTSIZE
Relative Address	0x00000160
Absolute Address	usb0: 0xE0002160 usb1: 0xE0003160
Width	17 bits
Access Type	rw
Reset Value	0x00001010
Description	This register controls the burst size used during data movement on the initiator/master interface.

Register BURSTSIZE Details

Field Name	Bits	Type	Reset Value	Description
TXPBURST (TX)	16:8	rw	0x10	<p>Programmable TX Burst Length.</p> <p>Default is the constant VUSB_HS_TX_BURST. This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus.</p> <p>If field AHBBRST of register SBUSCFG is different from zero, this field TXPBURST will return the value of the INCRx length.</p> <p>Supported values are integer values from 4 to 128. It is recommended to set this value to a integer sub-multiple of VUSB_HS_TX_CHAN. Different values will not use all the available buffer space, preventing proper TX endpoint priming in stream disable mode (SDIS bit of USBMODE register set to '1').</p>
RXPBURST (RX)	7:0	rw	0x10	<p>Programmable RX Burst Length. Default is the constant VUSB_HS_RX_BURST. This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory. If field AHBBRST of register SBUSCFG is different from zero, this field RXPBURST will return the value of the INCRx length.</p> <p>The supported values are integer values from 4 to 128. It is recommended to set this value to a integer sub-multiple of VUSB_HS_RX_DEPTH.</p>

Register ([usb](#)) TXFILLTUNING

Name	TXFILLTUNING
Software Name	TXFILL
Relative Address	0x00000164
Absolute Address	usb0: 0xE0002164 usb1: 0xE0003164
Width	22 bits
Access Type	mixed
Reset Value	0x00020000
Description	The fields in this register control performance tuning associated with how the Controller posts data to the TX latency FIFO before moving the data onto the USB bus.

Register TXFILLTUNING Details

Field Name	Bits	Type	Reset Value	Description
TXFIFOTHRES (BURST)	21:16	rw	0x2	<p>FIFO Burst Threshold:</p> <p>This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 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 under run 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 USBMODE register is set (SDIS).</p>
reserved	15:13	ro	0x0	RESERVED
TXSCHEALTH (HEALTH)	12:8	rw	0x0	<p>Scheduler Health Counter.</p> <p>This register increments when the 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.</p> <p>This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter. This counter will max. at 31.</p>
reserved	7	ro	0x0	RESERVED
TXSCHOH (OVERHEAD)	6:0	rw	0x0	<p>Scheduler Overhead.</p> <p>This register adds an additional fixed offset to the schedule time estimator described above as Tff. 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 that is too high for this register is not desired as it can needlessly reduce USB utilization.</p> <p>The time unit represented in this register is 1.267us when a device is connected in High-Speed Mode for OTG(on the go) & SPH(single port host) implementations.</p> <p>The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode for OTG & SPH implementations. The time unit represented in this register is always 1.267us for the MPH implementation</p>

Register ([usb](#)) TXTTFILLTUNING

Name	TXTTFILLTUNING
Relative Address	0x00000168
Absolute Address	usb0: 0xE0002168 usb1: 0xE0003168
Width	13 bits
Access Type	mixed
Reset Value	0x00000000
Description	This register provides a function similar to TXFILLTUNING except there is no equivalent to TXFIFOTHRES because the TT TX latency FIFO is always loaded in a single burst. Even

Register TXTTFILLTUNING Details

Field Name	Bits	Type	Reset Value	Description
TXTTSCHHEALTH	12:8	rw	0x0	TT Scheduler Health Counter Same description as TXSCHHEALTH
reserved	7:5	ro	0x0	RESERVED
TXTTSCHOH	4:0	rw	0x0	TT Scheduler Overhead Same description as TXSCHOH. The time unit represented in this register is 6.333us.

Register ([usb](#)) IC_USB

Name	IC_USB
Relative Address	0x0000016C
Absolute Address	usb0: 0xE000216C usb1: 0xE000316C
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	This register enable and controls the IC_USB FS/LS transceiver.

Register IC_USB Details

Field Name	Bits	Type	Reset Value	Description
IC8	31	ro	0x0	Inter-Chip transceiver enable 8. These bits enables the Inter-Chip transceiver for each port (for the MPH case). To enable the interface, the bits PTS must be set to '011b' in the PORTSC8. Writing a '1' to each bit selects the IC_USB interface for that port.
IC_VDD8	30:28	ro	0x0	Inter-Chip voltage selection 8 It selects which voltage is being supplied to the peripheral through each port This field is read-only and set to '000b' in case of device mode operation. This field is read-only and set to '000b' in case of Single port host controller.
IC7	27	ro	0x0	Inter-Chip transceiver enable 7. These bits enables the Inter-Chip transceiver for each port (for the MPH case). To enable the interface, the bits PTS must be set to '011b' in the PORTSC7. Writing a '1' to each bit selects the IC_USB interface for that port.
IC_VDD7	26:24	ro	0x0	Inter-Chip voltage selection 7 It selects which voltage is being supplied to the peripheral through each port This field is read-only and set to '000b' in case of device mode operation. This field is read-only and set to '000b' in case of Single port host controller.
IC6	23	ro	0x0	Inter-Chip transceiver enable 6. These bits enables the Inter-Chip transceiver for each port (for the MPH case). To enable the interface, the bits PTS must be set to '011b' in the PORTSC6. Writing a '1' to each bit selects the IC_USB interface for that port.
IC_VDD6	22:20	ro	0x0	Inter-Chip voltage selection 6 It selects which voltage is being supplied to the peripheral through each port This field is read-only and set to '000b' in case of device mode operation. This field is read-only and set to '000b' in case of Single port host controller.
IC5	19	ro	0x0	Inter-Chip transceiver enable 5. These bits enables the Inter-Chip transceiver for each port (for the MPH case). To enable the interface, the bits PTS must be set to '011b' in the PORTSC5. Writing a '1' to each bit selects the IC_USB interface for that port.

Field Name	Bits	Type	Reset Value	Description
IC_VDD5	18:16	ro	0x0	<p>Inter-Chip voltage selection 5</p> <p>It selects which voltage is being supplied to the peripheral through each port This field is read-only and set to '000b' in case of device mode operation.</p> <p>This field is read-only and set to '000b' in case of Single port host controller.</p>
IC4	15	ro	0x0	<p>Inter-Chip transceiver enable 4. These bits enables the Inter-Chip transceiver for each port (for the MPH case). To enable the interface, the bits PTS must be set to '011b' in the PORTSC4. Writing a '1' to each bit selects the IC_USB interface for that port.</p>
IC_VDD4	14:12	ro	0x0	<p>Inter-Chip voltage selection 4</p> <p>It selects which voltage is being supplied to the peripheral through each port This field is read-only and set to '000b' in case of device mode operation.</p> <p>This field is read-only and set to '000b' in case of Single port host controller.</p>
IC3	11	ro	0x0	<p>Inter-Chip transceiver enable 3. These bits enables the Inter-Chip transceiver for each port (for the MPH case). To enable the interface, the bits PTS must be set to '011b' in the PORTSC3. Writing a '1' to each bit selects the IC_USB interface for that port.</p>
IC_VDD3	10:8	ro	0x0	<p>Inter-Chip voltage selection 3</p> <p>It selects which voltage is being supplied to the peripheral through each port. This field is read-only and set to '000b' in case of device mode operation.</p> <p>This field is read-only and set to '000b' in case of Single port host controller.</p>
IC2	7	ro	0x0	<p>Inter-Chip transceiver enable 2. These bits enables the Inter-Chip transceiver for each port (for the MPH case). To enable the interface, the bits PTS must be set to '011b' in the PORTSC2. Writing a '1' to each bit selects the IC_USB interface for that port.</p>
IC_VDD2	6:4	ro	0x0	<p>Inter-Chip voltage selection 2</p> <p>It selects which voltage is being supplied to the peripheral through each port. This field is read-only and set to '000b' in case of device mode operation.</p> <p>This field is read-only and set to '000b' in case of Single port host controller.</p>

Field Name	Bits	Type	Reset Value	Description
IC1	3	rw	0x0	Inter-Chip transceiver enable 1. These bits enables the Inter-Chip transceiver for each port (for the MPH case). To enable the interface, the bits PTS must be set to '011b' in the PORTSC1. Writing a '1' to each bit selects the IC_USB interface for that port. If the Controller is not a MPH implementation, IC8 to IC2 will be '0' and Read-Only.
IC_VDD1	2:0	rw	0x0	Inter-Chip voltage selection 1 It selects which voltage is being supplied to the peripheral through each port 000 - No voltage 001 - 1.0V 010 - 1.2V 011 - 1.5V 100 - 1.8V 101 - 3.0V 110 - Reserved 111 - Reserved This field is read-only and set to '000b' in case of device mode operation. The voltage negotiation should happen between enabling port power (PP) in PORTSC1 register and asserting the run/stop bit in USBCMD register.

Register ([usb](#)) ULPI_VIEWPORT

Name	ULPI_VIEWPORT
Software Name	ULPIVIEW
Relative Address	0x00000170
Absolute Address	usb0: 0xE0002170 usb1: 0xE0003170
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	The register provides indirect access to the ULPI PHY register set. Although the core performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access.

Register ULPI_VIEWPORT Details

Field Name	Bits	Type	Reset Value	Description
ULPIWU (WU)	31	rw	0x0	ULPI Wakeup. Writing the '1' to this bit will begin the wakeup operation. The bit will automatically transition to '0' after the wakeup is complete. Once this bit is set, the driver can not set it back to '0'. Note: The driver must never execute a wakeup and a read/write operation at the same time.
ULPIRUN (RUN)	30	rw	0x0	ULPIRUN Writing the '1' to this bit will begin the read/write operation. The bit will automatically transition to '0' after the read/write is complete. Once this bit is set, the driver can not set it back to '0'. Note: The driver must never execute a wakeup and a read/write operation at the same time.
ULPIRW (RW)	29	rw	0x0	ULPI Read/Write Control'0' - Read.'1' - Write. This bit selects between running a read or write operation.
reserved	28	ro	0x0	Reserved
ULPISS (SS)	27	ro	0x0	ULPI Data Address. When a read or write operation is commanded, the address of the operation is written to this field.
ULPIPORT	26:24	rw	0x0	ULPI Port Number. For the wakeup or read/write operation to be executed, this value selects the port number to which a ULPI PHY is attached. The range is 0 to 7. This field should always be written '000b' for non MPH implementations.
ULPIADDR (ADDR)	23:16	rw	0x0	ULPI Data Address. When a read or write operation is commanded, the address of the operation is written to this field.
ULPIDATRD (DATRD)	15:8	ro	0x0	ULPI Data Read. After a read operation completes, the result is placed in this field.
ULPIDATWR (DATWR)	7:0	rw	0x0	ULPI Data Write. When a write operation is commanded, the data to be sent is written to this field

Register ([usb](#)) ENDPTNAK

Name	ENDPTNAK
Software Name	EPNAKISR
Relative Address	0x00000178

Absolute Address	usb0: 0xE0002178 usb1: 0xE0003178
Width	32 bits
Access Type	wtc
Reset Value	0x00000000
Description	Only implemented in device mode.

Register ENDPTNAK Details

Field Name	Bits	Type	Reset Value	Description
EPTN	31:16	wtc	0x0	TX Endpoint NAK Each TX endpoint has 1 bit in this field. The bit is set when the Controller sends a NAK handshake on a received IN token for the corresponding endpoint. Bit 15 - Endpoint #15 ... Bit 1 - Endpoint #1 Bit 0 - Endpoint #0 Only used in device mode.
EPRN	15:0	wtc	0x0	RX Endpoint NAK Each RX endpoint has 1 bit in this field. The bit is set when the Controller sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. Bit 15 - Endpoint #15 ... Bit 1 - Endpoint #1 Bit 0 - Endpoint #0 Only used in device mode.

Register ([usb](#)) ENDPTNAKEN

Name	ENDPTNAKEN
Software Name	EPNAKIER
Relative Address	0x0000017C
Absolute Address	usb0: 0xE000217C usb1: 0xE000317C
Width	32 bits
Access Type	rw
Reset Value	0x00000000
Description	Only implemented in device mode.

Register ENDPTNAKEN Details

Field Name	Bits	Type	Reset Value	Description
EPTNE	31:16	rw	0x0	TX Endpoint NAK enable 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. Bit 15 - Endpoint #15 ... Bit 1 - Endpoint #1 Bit 0 - Endpoint #0 Only used in device mode.
EPRNE	15:0	rw	0x0	RX Endpoint NAK enable 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. Bit 15 - Endpoint #15 ... Bit 1 - Endpoint #1 Bit 0 - Endpoint #0 Only used in device mode.

Register ([usb](#)) CONFIGFLAG

Name	CONFIGFLAG
Relative Address	0x00000180
Absolute Address	usb0: 0xE0002180 usb1: 0xE0003180
Width	32 bits
Access Type	ro
Reset Value	0x00000001
Description	This register is not used in this implementation. A read from this register returns a constant of a 00000001h to indicate that all port routings default to this host controller.

Register CONFIGFLAG Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:0	ro	0x1	RESERVED

Register ([usb](#)) PORTSC1

Name	PORTSC1
Software Name	PORTSCR1
Relative Address	0x00000184
Absolute Address	usb0: 0xE0002184 usb1: 0xE0003184
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	The Controller implement one The number of port registers implemented by a particular instantiation is documented in the HCSPARAM register. Software uses this information as an input parameter to determine how many ports need service. This implement contains only 1 host port.

Register PORTSC1 Details

Field Name	Bits	Type	Reset Value	Description
PTS	31:30	rw	0x0	This register bit pair is used in conjunction with the configuration constant VUSB_HS_PHY_TYPE to control which parallel transceiver interface is selected'010b' -> 2, 6 - ULPI Note that this field is made up from PORTSCx bits 25, 30 and 31.
STS	29	ro	0x0	Serial Transceiver Select This register bit is used in conjunction with the configuration constant VUSB_HS_PHY_SERIAL to control whether the parallel or serial transceiver interface is selected for FS and LS operation. The Serial Interface Engine can be used in combination with the UTMI+ physical interface to provide FS/LS signaling instead of the parallel interface. If VUSB_HS_PHY_SERIAL is set for 0 or 1 then this bit is read only. If VUSB_HS_PHY_SERIAL is 2 or 3 then this bit is read/write. This bit has no effect unless Parallel Transceiver Select is set to UTMI+. The Serial/1.1 and IC_USB physical interface will use the Serial Interface Engine for FS/LS signaling regardless of this bit value.
PTW	28	ro	0x0	Parallel Transceiver Width. Writing this bit to '0' selects the 8-bit [60MHz] UTMI+ interface.

Field Name	Bits	Type	Reset Value	Description
PSPD (PORTSCR_PSPD)	27:26	rw	0x0	Port Speed - RO. Default = 11b. This register field indicates the speed at which the port is operating. '00b' -> Full Speed '01b' -> Low Speed '10b' -> High Speed '11b' -> Not connected
PTS2	25	rw	0x0	Parallel Transceiver Select - RW. Default = Implementation dependent. MSB bit of PTS field.
PFSC (PORTSCR_PFSC)	24	rw	0x0	Port Force Full Speed Connect - RW. Default = 0b. Writing this bit to a '1b' will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with a HS host, hub or device. This bit is for debugging purposes.
PHCD (PORTSCR_PHCD)	23	ro	0x0	PHY Low Power Clock Disable - RW. Default = 0b. Writing this bit to a '1b' will disable the PHY clock. Writing a '0b' enables it. Reading this bit will indicate the status of the PHY clock. NOTE: The PHY clock cannot be disabled if it is being used as the system clock. In device mode, the PHY can be put into Low Power Clock Disable when the device is not running (USBCMD RS=0b) or the host has signaled suspend (PORTSCx SUSP=1b). Low Power Clock Disable will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the Controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low Power Clock Disable is completely under the control of software.
WKOC (PORTSCR_WKOC)	22	ro	0x0	Wake on Over-current Enable Writing '1' to this bit enables the port to be sensitive to over-current conditions as wakeup events. This field is zero if Port Power (PP) is '0' or in device mode. This bit is output from the controller as signal pwrctl_wake_ovrcurr_en for use by an external power control circuit. Only used in host mode.

Field Name	Bits	Type	Reset Value	Description
WKDS (PORTSCR_WKDS)	21	rw	0x0	<p>Wake on Disconnect Enable</p> <p>Writing this bit to a one enables the port to be sensitive to device disconnects as wakeup events. This field is zero if Port Power (PP) is '0' or in device mode. This bit is output from the controller as signal pwrctl_wake_dscntt_en for use by an external power control circuit.</p>
WKN (PORTSCR_WKN)	20	rw	0x0	<p>Wake on Connect Enable</p> <p>Writing this bit to a one enables the port to be sensitive to device connects as Wakeup events. This field is zero if Port Power(PP) is '0' or in device mode. This bit is output from the controller as signal pwrctl_wake_cnnt_en for use by an external power control circuit. Only used in host mode.</p>
PTC (PORTSCR_PTC)	19:16	rw	0x0	<p>Port Test Control</p> <p>Any other value than zero indicates that the port is operating in test mode.</p> <p>Value Specific Test</p> <p>0000b -> TEST_MODE_DISABLE</p> <p>0001b -> J_STATE</p> <p>0010b -> K_STATE</p> <p>0011b -> SE0 (host) / NAK (device)</p> <p>0100b -> Packet</p> <p>0101b -> FORCE_ENABLE_HS</p> <p>0110b -> FORCE_ENABLE_FS</p> <p>0111b -> FORCE_ENABLE_LS</p> <p>1000b -> Reserved</p> <p>... Reserved</p> <p>1111b -> Reserved</p>
PIC (PORTSCR_PIC)	15:14	rw	0x0	<p>Port Indicator Control</p> <p>Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is a zero. If P_INDICATOR bit is a one, then the bits are:</p> <p>Value Meaning</p> <p>00b -> Port indicators are off</p> <p>01b -> Amber</p> <p>10b -> Green</p> <p>11b -> Undefined</p> <p>Refer to the USB Specification Revision 2.0 for a description on how these bits are to be used. This field is output from the controller as signals port_ind_ctl_1 and port_ind_ctl_0 for</p>

Field Name	Bits	Type	Reset Value	Description
PO (PORTSCR_PO)	13	ro	0x0	Port Owner Port owner hand off is not implemented in this design, therefore this bit will always read back as 0b.
PP (PORTSCR_PP)	12	rw	0x0	Port Power The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC -> PP -> Operation 0b -> 0b -> Read Only. A Controller in device mode does not have port power control switches. 1b -> 1b/0b -> Read/Write. A Controller in host mode requires port power control switches. This bit represents the current setting of the switch ('0'=off, '1'=on). When power is not available on a port (i.e. PP equals to '0'), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the Controller driver from '1' to '0'(removing power from the port). In device mode port power control is not necessary, thus PPC and PP = 0.
LS (PORTSCR_LS)	11:10	ro	0x0	Line Status These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. The encoding of the bits are: 00b -> SE0 10b -> J-state 01b -> K-state 11b -> Undefined In host mode, the use of line state by the Controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of line state by the Controller driver is not necessary.
HSP (PORTSCR_HSP)	9	ro	0x0	High-Speed Port When the bit is one, the port is in high-speed mode and if set to zero, the port is not in a high-speed mode. Note: HSP is redundant with PSPD but will remain in the design for compatibility.

Field Name	Bits	Type	Reset Value	Description
PR (PORTSCR_PR)	8	rw	0x0	Port Reset - RW. Default = 0b. This field is zero if Port Power(PP) is '0'. Host mode: 1=Port is in Reset. 0=Port is not in Reset. Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register
SUSP (PORTSCR_SUSP)	7	rw	0x0	Suspend Host mode: 1=Port in suspend state. 0=Port not in suspend state. Port Enabled bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend Device mode: Read Only. 1=Port in suspend state. 0=Port not in suspend state. In device mode this bit is a read only status bit.
FPR (PORTSCR_FPR)	6	rw	0x0	Force Port Resume 1= Resume detected /driven on port. 0=No resume (K-state) detected /driven on port.
OCC (PORTSCR_OCC)	5	rw	0x0	Over-current Change This bit gets set to '1' when there is a change to Over-current Active. Software clears this bit by writing a '1' to this bit position. When in host mode implementations the user can provide over-current detection to the vbus_pwr_fault input for this condition. For device mode this bit shall always be '0'.
OCA (PORTSCR_OCA)	4	ro	0x0	Over-current Active Value Meaning '1b' -> This port currently has an over-current condition. '0b' -> This port does not have an over-current condition. This bit will automatically transition from '1' to '0' when the over current condition is removed. For host mode implementations the user can provide over-current detection to the vbus_pwr_fault input for this condition. For device mode implementations this bit shall always be '0'.

Field Name	Bits	Type	Reset Value	Description
PEC (PORTSCR_PEC)	3	wtc	0x0	<p>Port Enabled Change</p> <p>If set to '1' indicates a Port Enabled/Disabled status change.</p> <p>Host mode:</p> <p>For the root hub, this bit gets set to a '1' 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 '1' to it. This field is '0' if Port Power(PP) is '0'.</p> <p>Device mode:</p> <p>The device port is always enabled. (This bit will be '0').</p>
PE (PORTSCR_PE)	2	wtc	0x0	<p>Port Enabled</p> <p>1 -> Enable</p> <p>0-> Disable</p> <p>Host mode:</p> <p>Ports can only be enabled by Controller as a part of the reset and enable. Software cannot enable a port by writing a '1' to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the software. This field is '0' if Port Power(PP) is '0' in host mode.</p> <p>Device mode:</p> <p>The device port is always enabled. (This bit will be always '1').</p>

Field Name	Bits	Type	Reset Value	Description
CSC (PORTSCR_CSC)	1	wtc	0x0	<p>Connect Status Change</p> <p>If set to '1' indicates a change in Current Connect Status (CCS).</p> <p>Host mode:</p> <p>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, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a '1' to it. This field is '0' if Port Power(PP) is '0' in host mode.</p> <p>Device mode:</p> <p>This bit is undefined in device mode.</p>
CCS (PORTSCR_CCS)	0	ro	0x0	<p>Current Connect Status.</p> <p>Host mode:</p> <p>1 -> Device is present on port. 0 -> No device is present.</p> <p>Device mode:</p> <p>1 -> Attached. 0 -> Not Attached.</p>

Register ([usb](#)) OTGSC

Name	OTGSC
Software Name	OTGCSR
Relative Address	0x000001A4
Absolute Address	usb0: 0xE00021A4 usb1: 0xE00031A4
Width	32 bits
Access Type	mixed
Reset Value	0x00000020
Description	<p>The Controller implements one On-The-Go (OTG) Status and Control register.</p> <p>The OTGSC register has four sections:</p> <p>OTG Interrupt enables (Read/Write)</p> <p>OTG Interrupt status (Read/Write to Clear)</p> <p>OTG Status inputs (Read Only)</p> <p>OTG Controls (Read/Write)</p>

Register OTGSC Details

Field Name	Bits	Type	Reset Value	Description
reserved	31	ro	0x0	Reserved
DPIE (OTGSC_DPIE)	30	rw	0x0	Data Pulse Interrupt Enable This bit enables the generation of an interrupt if bit DPIS is set.
1msE (OTGSC_1MSE)	29	rw	0x0	1 millisecond timer Interrupt Enable This bit enables the generation of an interrupt if bit 1msS is set.
BSEIE (OTGSC_BSEE)	28	rw	0x0	B Session End Interrupt Enable This bit enables the generation of an interrupt if bit BSEIS is set
BSVIE (OTGSC_BSVIE)	27	rw	0x0	B Session Valid Interrupt Enable This bit enables the generation of an interrupt if bit BSVIS is set.
ASVIE (OTGSC_ASVIE)	26	rw	0x0	A Session Valid Interrupt Enable This bit enables the generation of an interrupt if bit ASVIS is set.
AVVIE (OTGSC_AVVIE)	25	rw	0x0	A VBus Valid Interrupt Enable This bit enables the generation of an interrupt if bit AVVIS is set.
IDIE (OTGSC_IDIE)	24	rw	0x0	USB ID Interrupt Enable This bit enables the generation of an interrupt if bit IDIS is set.
reserved	23	ro	0x0	Reserved
DPIS (OTGSC_DPIS)	22	wtc	0x0	Data Pulse Interrupt Status This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host ('11b') and PORTSC0.PP = Off ('0b'). Software must write a '1' to clear this bit.
1msS (OTGSC_1MSS)	21	wtc	0x0	1 millisecond timer Interrupt Status. This bit is set once every millisecond. Software must write a '1' to clear this bit.
BSEIS (OTGSC_BSEIS)	20	wtc	0x0	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 (OTGSC_BSVIS)	19	wtc	0x0	B Session Valid Interrupt Status This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software must write a '1' to clear this bit.

Field Name	Bits	Type	Reset Value	Description
ASVIS (OTGSC_ASVIS)	18	wtc	0x0	A Session Valid Interrupt Status This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software must write a '1' to clear this bit.
AVVIS (OTGSC_AVVIS)	17	wtc	0x0	Frame Index This bit is set when VBus has either 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 (OTGSC_IDIS)	16	wtc	0x0	USB ID Interrupt Status This bit is set when a change on the ID input has been detected. Software must write a '1' to clear this bit.
reserved	15	ro	0x0	Reserved
DPS (OTGSC_DPS)	14	ro	0x0	Data Bus Pulsing Status A '1' indicates data bus pulsing is being detected on the port.
1msT (OTGSC_1MST)	13	ro	0x0	1 millisecond timer toggle This bit toggles once per millisecond.
BSE (OTGSC_BSE)	12	ro	0x0	B Session End Indicates VBus is below the B session end threshold.
BSV (OTGSC_BSV)	11	ro	0x0	B Session Valid Indicates VBus is above the B session valid threshold.
ASV (OTGSC_ASV)	10	ro	0x0	A Session Valid Indicates VBus is above the A session valid threshold.
AVV (OTGSC_AVV)	9	ro	0x0	A VBus Valid. Indicates VBus is above the A VBus valid threshold.
ID (OTGSC_ID)	8	ro	0x0	USB ID'0' = A device, '1' = B device.
HABA (OTGSC_HABA)	7	rw	0x0	Hardware Assist B-Disconnect to A-connect'0' = Disabled, '1' = Enable automatic B-Disconnect to A-Connect sequence.
HADP (OTGSC_HADP)	6	rw	0x0	Hardware Assist Data-Pulse If set, the hardware assist data pulsing sequence starts.

Field Name	Bits	Type	Reset Value	Description
IDPU (OTGSC_IDPU)	5	rw	0x1	ID Pullup This bit provide control over the ID Pullup resister.'0' = off, '1' = on. When this bit is '0' the ID input will not be sampled.
DP (OTGSC_DP)	4	rw	0x0	Data Pulsing Setting this bit causes the pullup on DP to be asserted for data pulsing during SRP.
OT (OTGSC_OT)	3	rw	0x0	OTG Termination This bit must be set when the Controller is in device mode. It controls the pulldown on DM.
HAAR (OTGSC_HAAR)	2	rw	0x0	Hardware Assist Auto-Reset'0' = Disabled, '1' = Enable automatic reset after connect on host port.
VC (OTGSC_VC)	1	rw	0x0	VBUS Charge Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP.
VD (OTGSC_VD)	0	rw	0x0	VBUS Discharge. Setting this bit causes VBus to discharge through a resistor.

Register ([usb](#)) USBMODE

Name	USBMODE
Software Name	MODE
Relative Address	0x000001A8
Absolute Address	usb0: 0xE00021A8 usb1: 0xE00031A8
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	USB Mode Selection register

Register USBMODE Details

Field Name	Bits	Type	Reset Value	Description
SRT	15	rw	0x0	<p>Shorten Reset Time</p> <p>When the Controller is in host mode, this bit enables a bypass of the Chirp J/K reset handshake, saving 6-7ms in simulation time for each reset sequence. This bit should only be used for initial system integration simulations, and should always be set to 0 for normal operation.</p>
reserved	14:6	ro	0x0	Reserved
VBPS	5	rw	0x0	<p>Vbus Power Select'0' -> Output is '0' '1' -> Output is '1' 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.</p> <p>Only used in host mode.</p>
SDIS	4	rw	0x0	<p>Stream Disable Mode'0' -> Inactive'1' -> Active Device mode:</p> <p>Setting to a '1' disables double priming on both RX and TX for low bandwidth systems.</p> <p>Host Mode:</p> <p>Setting to a '1' ensures that overruns/under runs of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet.</p>
SLOM	3	rw	0x0	<p>Setup Lockout Mode</p> <p>This bit controls behavior of the setup lock mechanism.'0' -> Setup Lockouts On.'1' -> Setup Lockouts Off (DCD requires use of Setup Data Buffer Tripwire in USBCMD).</p> <p>Only used in device mode</p>

Field Name	Bits	Type	Reset Value	Description
ES	2	ro	0x0	<p>Endian Select</p> <p>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 -> 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.</p>
CM	1:0	rw	0x0	<p>Controller Mode</p> <p>Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host & device capability (OTG), the Controller will default to an idle state and will need to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RST bit in the USBCMD register before reprogramming this register.</p> <p>Value Meaning '00b' -> Idle (Default for combination host/device). '01b' -> Reserved. '10b' -> Controller in device mode (Default for device only controller). '11b' -> Controller in host mode (Default for host only controller).</p>

Register ([usb](#)) ENDPTSETUPSTAT

Name	ENDPTSETUPSTAT
Software Name	EPSTAT
Relative Address	0x000001AC
Absolute Address	usb0: 0xE00021AC usb1: 0xE00031AC
Width	16 bits
Access Type	wtc
Reset Value	0x00000000
Description	Only implemented in device mode.

Register ENDPTSETUPSTAT Details

Field Name	Bits	Type	Reset Value	Description
ENDPTSETUPSTAT	15:0	wtc	0x0	<p>ENDPTSETUPSTATSetup Endpoint Status</p> <p>For every setup transaction that is received, the corresponding bit in this register is set to '1'. Software must clear to acknowledge the setup transfer by writing a '1' to the 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 if the setup lockout mechanism is engaged.</p> <p>Only used in device mode.</p>

Register ([usb](#)) ENDPTPRIME

Name	ENDPTPRIME
Software Name	EPPRIME
Relative Address	0x000001B0
Absolute Address	usb0: 0xE00021B0 usb1: 0xE00031B0
Width	32 bits
Access Type	wtc
Reset Value	0x00000000
Description	Only implemented in device mode

Register ENDPTPRIME Details

Field Name	Bits	Type	Reset Value	Description
PETB	31:16	wtc	0x0	<p>Prime Endpoint Transmit Buffer - RWS. Default = 0000h.</p> <p>For each endpoint a corresponding bit is used to request that a buffer prepared for a transmit operation in order 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 will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed.</p> <p>Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.</p> <p>Bit 15 - Endpoint #15</p> <p>...</p> <p>Bit 1 - Endpoint #1</p> <p>Bit 0 - Endpoint #0</p> <p>Only used in device mode.</p>
PERB	15:0	wtc	0x0	<p>Prime Endpoint Receive Buffer</p> <p>For each endpoint, a corresponding bit is used to request a buffer prepare for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a '1' to the corresponding bit whenever posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed.</p> <p>Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.</p> <p>Bit 15 - Endpoint #15</p> <p>...</p> <p>Bit 1 - Endpoint #1</p> <p>Bit 0 - Endpoint #0</p> <p>Only used in device mode.</p>

Register ([usb](#)) ENDPTFLUSH

Name	ENDPTFLUSH
Software Name	EPFLUSH
Relative Address	0x000001B4
Absolute Address	usb0: 0xE00021B4 usb1: 0xE00031B4
Width	32 bits
Access Type	wtc
Reset Value	0x00000000
Description	The Flush operation for an endpoint will clear the Ready status of that endpoint and re-align the Latency Buffer pointers, but not clear the actual data that resides in the Latency Buffers.

Register ENDPTFLUSH Details

Field Name	Bits	Type	Reset Value	Description
FETB	31:16	wtc	0x0	<p>Flush Endpoint Transmit Buffer</p> <p>Writing a '1' to a bit(s) in this register will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful.</p> <p>Bit 15 - Endpoint #15</p> <p>...</p> <p>Bit 1 - Endpoint #1</p> <p>Bit 0 - Endpoint #0</p> <p>Only used in device mode.</p>
FERB	15:0	wtc	0x0	<p>Flush Endpoint Receive Buffer</p> <p>Writing a '1' to a bit(s) will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful.</p> <p>Bit 15 - Endpoint #15</p> <p>...</p> <p>Bit 1 - Endpoint #1</p> <p>Bit 0 - Endpoint #0</p> <p>Only used in device mode.</p>

Register ([usb](#)) ENDPTSTAT

Name	ENDPTSTAT
Software Name	EPRDY
Relative Address	0x000001B8
Absolute Address	usb0: 0xE00021B8 usb1: 0xE00031B8
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Only implemented in device mode.

Register ENDPTSTAT Details

Field Name	Bits	Type	Reset Value	Description
ETBR	31:16	ro	0x0	<p>Endpoint Transmit Buffer Ready</p> <p>One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a '1' by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register.</p> <p>Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated.</p> <p>Bit 15 - Endpoint #15</p> <p>...</p> <p>Bit 1 - Endpoint #1</p> <p>Bit 0 - Endpoint #0</p> <p>Only used in device mode.</p>
ERBR	15:0	ro	0x0	<p>Endpoint Receive Buffer Ready</p> <p>One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a '1' by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register.</p> <p>Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated.</p> <p>Bit 15 - Endpoint #15</p> <p>...</p> <p>Bit 1 - Endpoint #1</p> <p>Bit 0 - Endpoint #0</p> <p>Only used in device mode.</p>

Register ([usb](#)) ENDPTCOMPLETE

Name	ENDPTCOMPLETE
Software Name	EPCOMPL
Relative Address	0x000001BC
Absolute Address	usb0: 0xE00021BC usb1: 0xE00031BC
Width	32 bits
Access Type	ro
Reset Value	0x00000000
Description	Only implemented in device mode.

Register ENDPTCOMPLETE Details

Field Name	Bits	Type	Reset Value	Description
ETCE	31:16	ro	0x0	<p>Endpoint Transmit Complete Event</p> <p>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, then this bit will be set simultaneously with the USBINT. Writing a '1' will clear the corresponding bit in this register.</p> <p>Bit 15 - Endpoint #15</p> <p>...</p> <p>Bit 1 - Endpoint #1</p> <p>Bit 0 - Endpoint #0</p> <p>Only used in device mode.</p>
ERCE	15:0	ro	0x0	<p>Endpoint Receive Complete Event</p> <p>Each bit indicates a received event (OUT) 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, then this bit will be set simultaneously with the USBINT. Writing a '1' will clear the corresponding bit in this register.</p> <p>Bit 15 - Endpoint #15</p> <p>...</p> <p>Bit 1 - Endpoint #1</p> <p>Bit 0 - Endpoint #0</p> <p>Only used in device mode.</p>

Register ([usb](#)) ENDPTCTRL0

Name	ENDPTCTRL0
Software Name	EPCR0
Relative Address	0x000001C0
Absolute Address	usb0: 0xE00021C0 usb1: 0xE00031C0
Width	24 bits
Access Type	mixed
Reset Value	0x00800080
Description	Every device will implement Endpoint0 as a control endpoint. Only implemented in device mode.

Register ENDPTCTRL0 Details

Field Name	Bits	Type	Reset Value	Description
TXE (EPCR_TXE)	23	ro	0x1	TX Endpoint Enable Endpoint0 is always enabled and this bit always set to '1'.
reserved	22:20	ro	0x0	Reserved
TXT (EPCR_TXT_INTR)	19:18	ro	0x0	TX Endpoint Type Endpoint0 is fixed as a Control End Point and this field always set to '00'.
reserved	17	ro	0x0	Reserved
TXS (EPCR_TXS)	16	rw	0x0	TX Endpoint Stall Value Meaning '0' -> Endpoint OK. '1' -> Endpoint Stalled
reserved	15:8	ro	0x0	Reserved
RXE (EPCR_RXE)	7	ro	0x1	RX Endpoint Enable Endpoint0 is always enabled and this bit always set to '1'.
reserved	6:4	ro	0x0	Reserved
RXT (EPCR_RXT_INTR)	3:2	ro	0x0	RX Endpoint Type Endpoint0 is fixed as a Control End Point and this field always set to '00'.

Field Name	Bits	Type	Reset Value	Description
reserved	1	ro	0x0	Reserved
RXS (EPCR_RXS)	0	rw	0x0	<p>RX Endpoint Stall</p> <p>Value Meaning '0' -> Endpoint OK. '1' -> Endpoint Stalled</p> <p>Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request.</p>

Register ([usb](#)) ENDPTCTRL1

Name	ENDPTCTRL1
Relative Address	0x000001C4
Absolute Address	usb0: 0xE00021C4 usb1: 0xE00031C4
Width	24 bits
Access Type	mixed
Reset Value	0x00000000
Description	This is an ENDPTCTRL1 register for the endpoint 1 device. Only implemented in device mode.

Note: This register is the first in an array of 12 identical registers listed in the table below. The details provided in this section apply to the entire array.

Name	Address
ENDPTCTRL1	0xe00021c4
ENDPTCTRL2	0xe00021c8
ENDPTCTRL3	0xe00021cc
ENDPTCTRL4	0xe00021d0
ENDPTCTRL5	0xe00021d4
ENDPTCTRL6	0xe00021d8
ENDPTCTRL7	0xe00021dc
ENDPTCTRL8	0xe00021e0
ENDPTCTRL9	0xe00021e4
ENDPTCTRL10	0xe00021e8
ENDPTCTRL11	0xe00021ec
ENDPTCTRL12	0xe00021f0

Register ENDPTCTRL1 to ENDPTCTRL12 Details

Field Name	Bits	Type	Reset Value	Description
TXE	23	ro	0x0	TX Endpoint Enable Value Meaning'0' -> Disabled'1' -> Enabled An endpoint should be enabled only after it has been configured.
TXR	22	ro	0x0	TX Data Toggle Reset Write '1' will reset the PID sequence. Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
TXI	21	ro	0x0	TX Data Toggle Inhibit Value Meaning'0' -> PID Sequencing Enabled'1' -> PID Sequencing Disabled
reserved	20	ro	0x0	Reserved
TXT	19:18	rw	0x0	TX Endpoint Type'00' -> Control'01' -> Isochronous'10' -> Bulk'11' -> Interrupt
TXD	17	rw	0x0	TX Endpoint Data 0 Dual Port Memory Buffer/DMA Engine Should always be written as '0'.
TXS	16	rw	0x0	TX Endpoint Stall Value Meaning'0' -> Endpoint OK.'1' -> Endpoint Stalled
reserved	15:8	ro	0x0	Reserved
RXE	7	rw	0x0	RX Endpoint Enable'0' -> Disabled'1' -> Enabled An endpoint should be enabled only after it has been configured.
RXR	6	rw	0x0	RX Data Toggle Reset Write '1' will reset the PID sequence. Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
RXI	5	rw	0x0	RX Data Toggle Inhibit'0' -> PID Sequencing Enabled'1' -> PID Sequencing Disabled
reserved	4	rw	0x0	Reserved
RXT	3:2	rw	0x0	RX Endpoint Type Endpoint0 is fixed as a Control End Point and this field always set to '00'.

Field Name	Bits	Type	Reset Value	Description
RXD	1	rw	0x0	RX Endpoint Data Sink Value Meaning 0 Dual Port Memory Buffer/DMA Engine Should always be written as '0'.
RXS	0	rw	0x0	RX Endpoint Stall Value Meaning '0' -> Endpoint OK. '1' -> Endpoint Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request.

Access Types Legend

Access Type	Description
clonrd	Readable, clears value on read
clonwr	Readable, clears value on write
nsnsro	During non-secure access, if thread is non-secure, it is read only
nsnsrw	During non-secure access, if thread is non-secure, it is read write
nsnswo	During non-secure access, if thread is non-secure, it is write only
nssraz	During non-secure access, if thread is secure, it is read as zero
raz	Read as zero
ro	Read-only
rs	w: no effect, r: sets all bits
rud	Read undefined
rw	Normal read/write
rwso	Read/write, set only
sro	During secure access, it is read only
srw	During secure access, it is read write
swo	During secure access, it is write only
w0c	w: 1/0 no effect on/clears matching bit, r: no effect
w0crs	w: 1/0 no effect on/clears matching bit, r: sets all bits
w0s	w: 1/0 no effect on/sets matching bit, r: no effect
w0src	w: 1/0 no effect on/sets matching bit, r: clears all bits
w0t	w: 1/0 no effect on/toggles matching bit, r: no effect
w1	w: first one after ~hard~ reset is as-is, other w have no effects, r: no effect

Access Type	Description
w1crs	w: 1/0 clears/no effect on matching bit, r: sets all bits
w1src	w: 1/0 sets/no effect on matching bit, r: clears all bits
w1t	w: 1/0 toggles/no effect on matching bit, r: no effect
waz	Write as zero
wcrs	w: clears all bits, r: sets all bits
wo	Write-only
wo1	w: first one after ~hard~ reset is as-is, other w have no effects, r: error
woc	w: clears all bits, r: error
wos	w: sets all bits, r: error
wrc	w: as-is, r: clears all bits
wrs	w: as-is, r: sets all bits
ws	w: sets all bits, r: no effect
wsrc	w: sets all bits, r: clears all bits
wtc	Readable, write a 1 to clear
z	Access (read or write) as zero